UNIVERSITY OF VERONA

Department of Computer Science

TECHNICAL REPORT

Computer Vision Project

Automatic Camera Pose Estimation

Hui Shi

Curriculum Computer Engineering for Robotics and Smart Industry

Verona 2023

Table of Contents

1. Introduction

The purpose of this report is to cover the technical details involved in the software development of the Automatic Camera Pose Estimation Project. It provides sufficient technical details to allow the user to use, improve, and undertake further development of the software.

Exterior orientation estimation methods (e.g. Fiore [1], Lowe) are widely used in camera pose(extrinsic parameters) estimation. However these methods require knowing the 3D -> 2D correspondences of the scene as shown in Figure 1, which are not always available. The developed software package provides an algorithm estimating the 3D -> 2D correspondences. The algorithm first takes a 3D model and a picture of a calibrated scene as input, and matches the projected 3D points with the 2D feature points computed using SIFT (Scale Invariant Feature Transform) method. The matches are taken as reference points. The algorithm then takes a target image whose camera pose to be estimated as input and computes its 2D feature points. These 2D feature points are matched with the reference points by comparing their descriptors, obtaining the 3D -> 2D correspondences of the target image. Finally, the program runs the Fiore's method and RANSAC (Random Sample Consensus) fitting to estimate the camera pose of the target image. Experimental validation shows that the algorithm performs well on camera pose estimation.

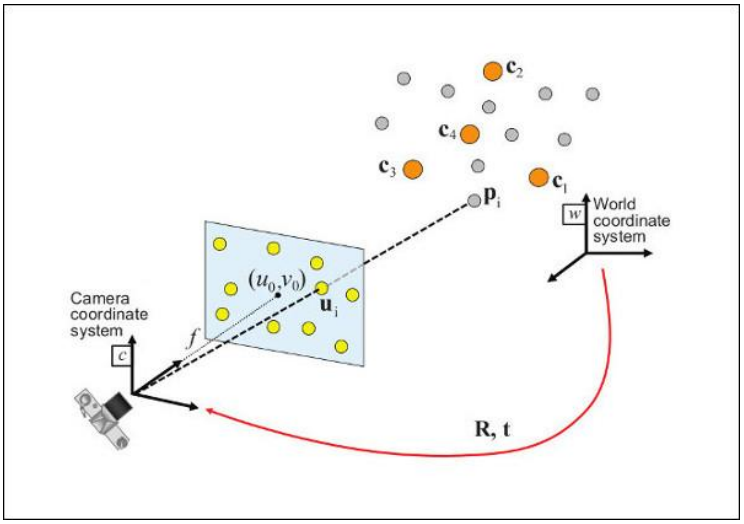


Figure 1. *Exterior orientation estimation*

2. Requirements

2.1 Functional Requirements

- Extract 2D feature points from the reference image, using SIFT functionality from the VLFeat library [2].
- Project 3D points to the reference image.

- Match the largest 2D feature points with the projected 3D points, and take the matches as reference points.
- Extract 2D feature points from the target image.
- Match the 2D feature points of the target image with the reference points, by comparing their descriptors.
- Estimate the camera pose of the target image using Fiore's exterior estimation method, which takes 3D-->2D correspondences as input. Robust fitting using RANSAC.

2.2 Non-Functional Requirements

- Microsoft Windows 10
- MATLAB R2021b
- VLFeat 0.9.21

3. Design

The algorithm is implemented as MATLAB programs in the files extract_reference_points.m and camera_pose_estimator.m in the developed software package.

3.1 Input and Output

Input:

- A 3D model and a reference picture of a calibrated scene, and its calibration data
- A target image whose camera pose to be estimated and its camera intrinsic parameters

Output: Camera pose(extrinsic parameters) of the target image

3.2 Algorithm

The extraction of the reference points is implemented in the file extract_reference_points.m containing 3 steps described in section 3.2.1, 3.2.2, 3.2.3. The camera pose estimation of the target image is implemented in the file camera_pose_estimator.m containing 3 steps described in section 3.2.4, 3.2.5, 3.2.6.

3.2.1 Extract 2D feature points from the reference image

The extraction of the 2D feature points from the reference image used the vl_sift function (SIFT method) from the VLFeat library as shown in Figure 2. The vl_sift function returns the frames(keypoints) and descriptors of the feature points. The frames matrix has a column for each frame. The first two rows are the center of the frame, the third row is the scale , and the forth row is the orientation. A SIFT descriptor is a 3-D spatial histogram of the image gradients in characterizing the appearance of a keypoint [3]. Each descriptor has 128 elements. After the extraction, the scale of the extracted frames are then sorted and the biggest frames are taken to be used to match with the 3D points. The algorithm adjust the number of the frames considered as the biggest frames depending on the size of the image. For instance, from the author's experiments, bigger images generates more feature points. Images with size 2448*3264 and 3648*5472 generates more than 20,000 and more than 70,000 feature points respectively, and an image with size 480*640 generates 1,500 to 2,000 feature points. The selected feature points are plotted by the function vl_plotframe() [2]. The result is shown on Figure 3 as yellow frames.

extract_reference_points.m

```
% 1. Extract 2D feature points from the reference image, using SIFT from the vlfeat lib.
% ref : https://www.vlfeat.org/overview/sift.html#tut.sift.match
img_ref = rgb2gray(img_ref) ;
img_ref = im2single(img_ref);
[frame_ref, descriptor_ref] = vl_sift(img_ref) ;

% sort the scale of the extracted frames and take the biggest ones
scale_f_ref = frame_ref(3,:);
[dump,scaleind_ref] = sort(scale_f_ref,'descend');
% change the num_feature (number of biggest frames) if the image size is too small
if size(img_ref,1) <= 1000 && size(img_ref,2) <= 1000
    num_feature = 1000;
else
    num_feature = 5000;
end
scaleind_ref = scaleind_ref(:,1:num_feature)';
frame_ref = frame_ref(:,scaleind_ref);
descriptor_ref = descriptor_ref(:,scaleind_ref);

% show the selected feature points
figure('Name','Reference points')
imshow(img_ref_RGB);
hold on
h_ref = vl_plotframe(frame_ref) ;
set(h_ref,'color','y','linewidth',2) ;
```

Figure 2. *Implementation of extracting 2D feature points from the reference image*

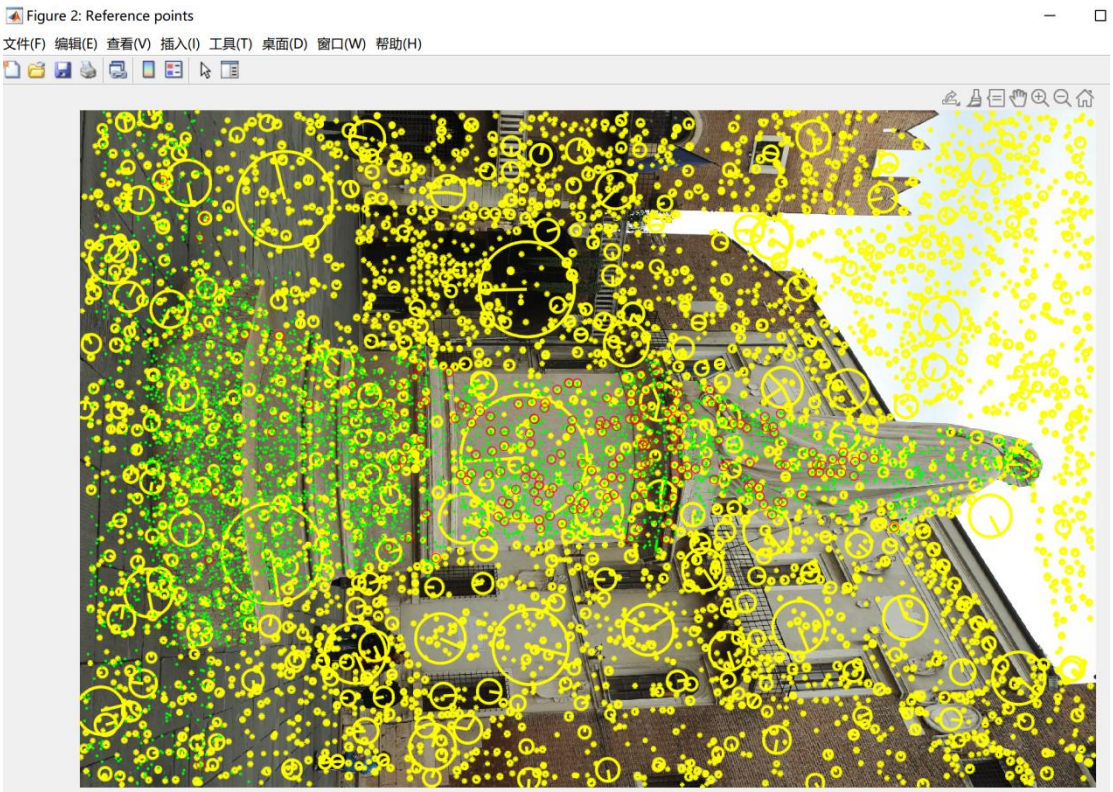


Figure 3. *Selected 2D feature points(yellow), projected 3D points(green), and their matches(red) as the reference points*

3.2.2 Project 3D points to the reference image

The projection of the 3D points to the reference image is done by the proj() function in the developed software package. The projected 3D points are shown on Figure 3 as green dots.

3.2.3 Match the largest 2D feature points with the projected 3D points

As the projection of the 3D points might has small error which cause it might lie around the 2D feature points. The algorithm matches the selected 2D feature points with the projected 3D points by searching around the feature points with a window as shown in Figure 4. The matches are taken as reference points and their corresponding descriptors are saved in reference_points.mat to be used by camera_pose_estimator.m. The matches are shown in Figure 3 as red circles.

```
% 3. Match the largest 2D feature points with the 3D points, and take the
% matches as reference points
window_size=3; % change the window size if necessary
ind_match_3D = [];
ind_match_feature = [];
% search around the feature points, find the cooresponding projected 3D points
for i=1:size(frame_ref,2) % for every feature point
    for j =1:size(u) % for every projected u
        %  if u is within x +/- window_size
        if frame_ref(1,i) >= round(u(j))-window_size && frame_ref(1,i) <= round(u(j))+ window_size
            % if v is within y +/- window_size
            if frame_ref(2,i) >= round(v(j))-window_size && frame_ref(2,i) <= round(v(j))+ window_size
                % record the matches
                ind_match_3D = [ind_match_3D j];
                ind_match_feature = [ind_match_feature i];
            end
        end
    end
end
% show the reference points as red circles
plot(u(ind_match_3D(1,:)), v(ind_match_3D(1,:)), 'ro');

ref_feature = frame_ref(:,ind_match_feature(1,:));
ref_p3D = point_3D(ind_match_3D(1,:), :);
% assign to reference points the 2D descriptor inherited from the 2D feature points
ref_descriptor = descriptor_ref(:,ind_match_feature(1,:));

% save the ref_feature, ref_p3D, ref_descriptor, num_feature to reference_points.mat
% to be used by camera_pose_estimator.m
save('reference_points.mat', "ref_p3D", "ref_feature", "ref_descriptor", "num_feature");
```

Figure 4. *Implementation of matching the largest 2D feature points with the projected 3D points*

3.2.4 Extract 2D feature points from the target image.

The extraction of the 2D feature points from the target image used the vl_sift function from the VLFeat library. The algorithm then sort the scale of the extracted frames and take the biggest ones (same number as the frames taken from the reference image) to be used to match with the reference points. The selected feature points are shown in Figure 5 as yellow frames.

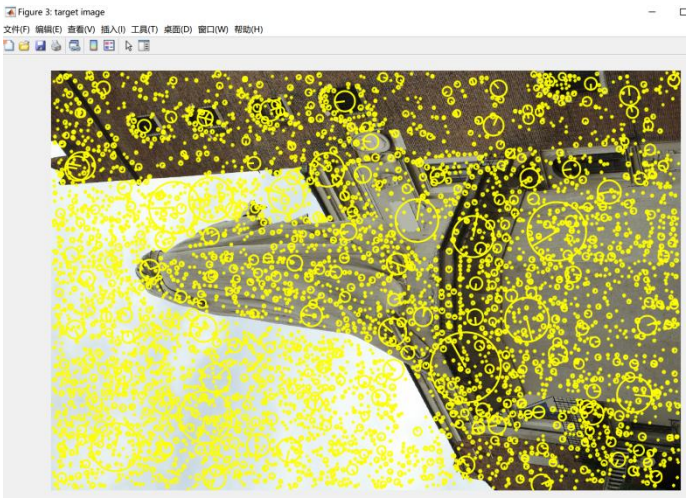


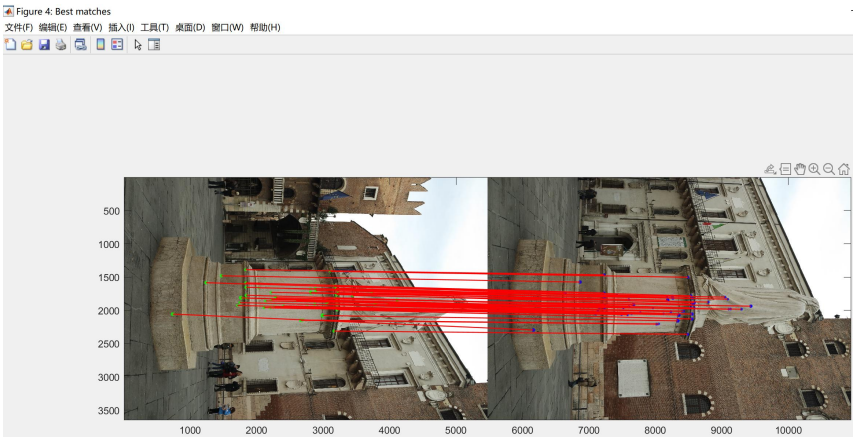
Figure 5. *Selected 2D feature points of the target image*

3.2.5 Match the 2D feature points of the target image with the reference points

The matching is done by comparing the descriptors of the reference points and the descriptors of the feature points from the target image, using the vl_ubcmatch() [2] function as shown in Figure 6. For each descriptor in the reference set, vl_ubcmatch() finds the closest descriptor in the target set (as measured by the L2 norm of the difference between them) [2],[3]. The function returns matches and scores. The first row of matches stores the index of the reference descriptor and the second row stores the index of the target descriptor. The distance between the pair is stored in scores. The algorithm then sorts the scores and take the best matches with a certain ratio, obtaining the 3D -> 2D correspondences needed for the Fiore's exterior estimation. The best matches are shown in Figure 7 with blue lines linking them.

```
% 2. Match the 2D feature points of the target image with the reference points,
% by comparing their descriptors
[matches, scores] = vl_ubcmatch(ref_descriptor, descriptor_target) ;
% sort the scores and take the best matches
[dump,scoreind] = sort(scores,'ascend');
score_ratio = 0.5; % change the score_ratio if necessary
best_matchesind = scoreind(1,1:round(size(scoreind,2)*score_ratio));
% obtain 3D-->2D correspondences
best_match_3D = ref_p3D(matches(1,best_matchesind),:);
best_match_2D = frame_target(:,matches(2,best_matchesind));

% show the best matches
% combine reference image and target image
newfig=zeros(size(img_ref,1), size(img_ref,2)+size(img_target,2),3);
newfig(:,1:size(img_ref,2),:) = img_ref_RGB;
newfig(1:size(img_target,1) ,(size(img_ref,2)+1):end,:)=img_target_RGB;
newfig=uint8(newfig);
figure('Name','Best matches');
image(newfig);
axis image;
hold on
% plot the best matches
f_targ_shifted = frame_target;
f_targ_shifted(1,:) = f_targ_shifted(1,:)+size(img_ref,2); % shift the target image
h1 = vl_plotframe(ref_feature(:,matches(1,best_matchesind)));
h2 = vl_plotframe(f_targ_shifted(:,matches(2,best_matchesind)));
set(h1,'color','g','linewidth',2);
set(h2,'color','b','linewidth',2);
hold on
% plot lines between the best matches
for i = 1:size(best_matchesind,2)
    indx = best_matchesind(i);
    line([ref_feature(1,matches(1,indx)) f_targ_shifted(1,matches(2,indx))],...
    [ref_feature(2,matches(1,indx)) f_targ_shifted(2,matches(2,indx))], 'linewidth',1, 'color','r')
end
```

Figure 6. *Implementation of matching the 2D feature points of the target image with the reference points*



(a)

(b)

Figure 7. *Best matches between the feature points of the target image and the reference (a)no outliers, (b) occurrence of outliers*

3.2.6 Camera pose estimation using Fiore's exterior estimation and RANSAC fitting

The camera pose estimation of the target image is done by the ransac_fiore() function as shown in Figure 8 (a). It takes the 3D -> 2D correspondences and the intrinsic camera parameters of the target image as input and returns the estimated camera pose (extrinsic parameters). As shown in Figure 8 (b), the ransac_fiore() function first takes the Fiore's method as the model to be fitted and estimates its parameters using minimum number of correspondences requested. The estimated model parameters are tested on the rest of points and the inliers are selected within a certain distance between the 2D points and the projected 3D points. After a predefined number of iterations, the model parameters estimated with the highest score (number of inliers) is obtained as the best model parameters. The function finally runs the Fiore's method using the inliers corresponding to the best model parameters, obtaining the final camera pose.

The estimated camera pose is tested by a comparison of the projections using the estimated extrinsic parameters and the known one, as shown in Figure 9. The estimated pose and the known pose are printed in the command line window as a comparison as shown in Figure 9.

```
% 3. Estimate the camera pose of the target image using Fiore's exterior
% estimation method, which takes 3D-->2D correspondences as input. Robust fitting using RANSAC.
% read the camera parameters
name_targ_xmp = [name_target(1:end-3) 'xmp'];
[K1, R1, t1] = read_xmp(name_targ_xmp);
% the known camera extrinsic parameter and ppm are used for later comparing
% with the estimated result.
G1 = [R1 t1; 0 0 0 1];
ppm_known = K1 * [1 0 0 0
        0 1 0 0
        0 0 1 0] * G1;

% run Fiore's method and robust fitting using RANSAC, obtain the estimated camera pose
tollerance = 10; % acceptable distance used to filter out the outliers
max_iterations = 100; % change the RANSAC iteration times if needed
[G_estimated, inliers, outliers] = ransac_fiore(K1, best_match_3D', best_match_2D(1:2,:), tollerance, max_iterations);
P_estimated=K1*G_estimated;

% project the 3D points to the image using the estimated ppm (red circle)
% and compare with the projection using the known ppm (blue dot)
figure('Name','projection of 3D points using estimated ppm "red o" and known ppm "blue ."');
[u2,v2] = proj(ppm_known,point_3D);
[u3,v3] = proj(P_estimated,point_3D);
imshow(img_target_RGB);
hold on
plot(u2,v2,'b.');
plot(u3,v3,'ro')
```

(a)

```
function [G, inliers, outliers] = ransac_fiore(K, points_3D, points_2D, tollerance, max_iterations)

inliers = [];
outliers = [];
p_model = 6; % minimum number of points needed for Fiore's exterior estimation

i = 0;
while (i < max_iterations) % iterate max_iterations times
    inliers_temp = [];
    outliers_temp = [];
    % make the data to be random
    rand_ind = randperm(size(points_3D,2));
    points_3D_rand = points_3D(:,rand_ind);
    points_2D_rand = points_2D(:,rand_ind);

    % run Fiore's exterior estimation, obtain the model parameters
    [G_temp,s] = exterior_fiore(K,points_3D_rand(:,1:p_model),points_2D_rand(:,1:p_model));

    % project the rest of the 3D points to the image using the estimated model parameters
    p_3D = points_3D_rand(:,p_model+1:end);
    ppm_temp=K*G_temp;
    [u,v] = proj(ppm_temp,p_3D');
    projected_3D = [u';v'];
    % take the rest of the 2D points
    p_2D = points_2D_rand(:,p_model+1:end);


    % model fitting using the rest of the data
    % filter out the outliers using the distance between the 2D points and projected 3D points
    for j = 1:size(projected_3D,2)
        distance = norm(p_2D(:,j)-projected_3D(:,j));
        if (abs(distance) <= tollerance)
            inlier = [p_3D(:,j); p_2D(:,j)];
            inliers_temp = [inliers_temp inlier];
        else
            outlier = [p_3D(:,j); p_2D(:,j)];
            outliers_temp = [outliers_temp outlier];
        end
    end
    % update the output if there is a better camera pose estimation
    % according to the number of inliers
    if (size(inliers_temp, 2) > size(inliers, 2))
        inliers = inliers_temp;
        outliers = outliers_temp;
        G = G_temp;
    end
    i = i + 1;
end

% run Fiore's exterior estimation, obtain the camera pose using all inliers from the best
% model parameters estimation
[G,s] = exterior_fiore(K,inliers(1:3,:),inliers(4:5,:));
```

(b)

Figure 8. *(a) Implementation of estimating the camera pose of the target image (b) Implementation of* the ransac_fiore() function

(a)

```
known camera pose:
ans =
    0.4825   -0.1275   -0.8665   64.8652
    0.1719    0.9839   -0.0490  -60.6174
    0.8588   -0.1253    0.4967  -29.7142
estimated camera pose:
G_estimated =
    0.4773   -0.1245   -0.8699   64.9583
    0.1680    0.9846   -0.0488  -60.6419
    0.8625   -0.1228    0.4909  -29.4892
```

(b)

Figure 9. *(a)Projection of the 3D points using estimated ppm(red) and known ppm(blue) (b)Printed estimated camera pose and known camera pose*

4. Testing and Validation

The developed algorithm was tested on two sets of images with different image sizes, a subset from the Dante Statue data set and an image data set of the Ca' Vignal 2 building of the University of Verona, both provided by the professor. The images from the Dante Statue data set has size 3648*5472 and the image of the Ca' Vignal 2 building has size 2448*3264.

4.1 Experiment 1 - Dante Statue

Input:

The image _SAM1001.JPG from the Dante Statue data set and the corresponding visible 3D points of the statue were taken as the reference, as shown in Figure 10.

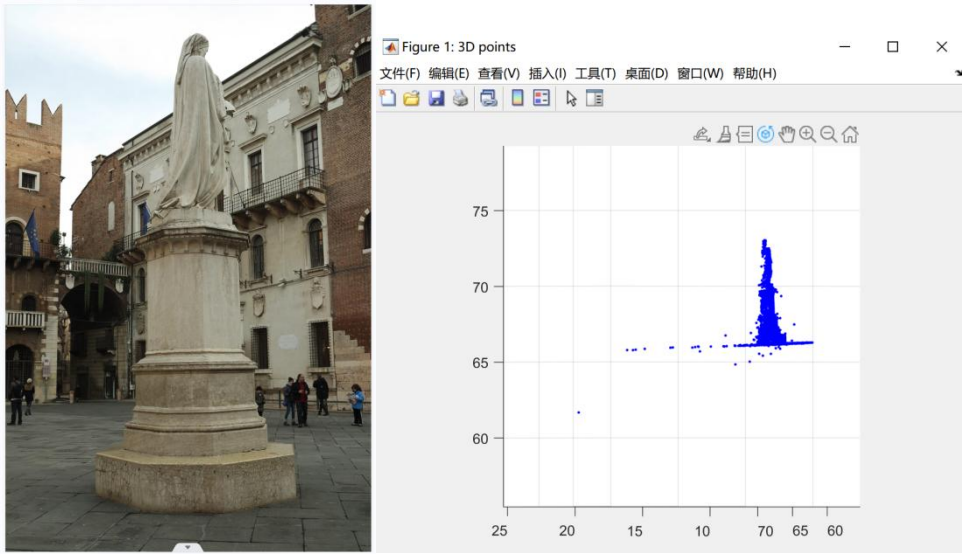


Figure 10. *Reference image _SAM1001.JPG and corresponding 3D points*

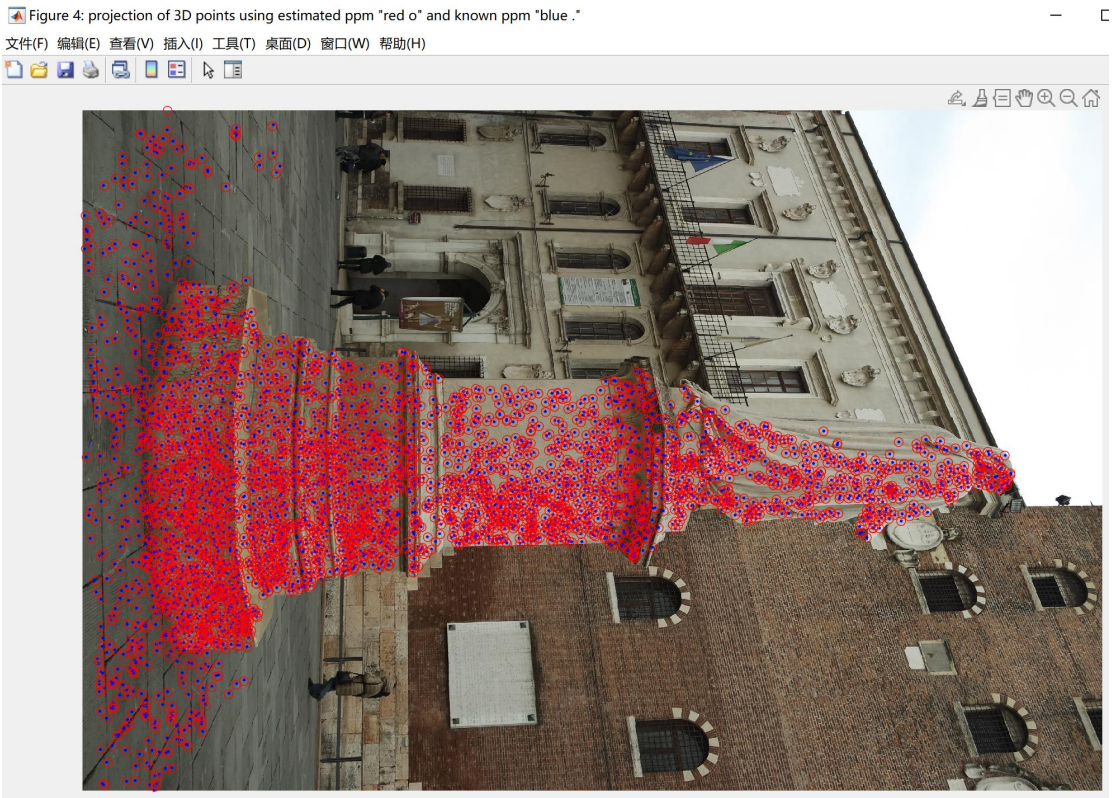
The following five images which share most of the same visible 3D points of the statue with the reference image, were taken as target images. The camera orientation and position of these images are different from the reference, as shown in Figure 11.

_SAM1002.JPG _SAM1079.JPG _SAM1080.JPG

_SAM1107.JPG _SAM1108.JPG

Figure 11. *Target images*

Result:
For each target image, the estimated camera pose was tested by a comparison of the projections of the 3D points onto the target image using the estimated pose and the known one. The projections and the comparison of the estimated pose and the known pose are shown in Figure 12. The algorithm was able to filter out the outliers occurred in the image _SAM1080.JPG and performs well on the camera pose estimation, as shown in Figure 12.



known camera pose:

ans =

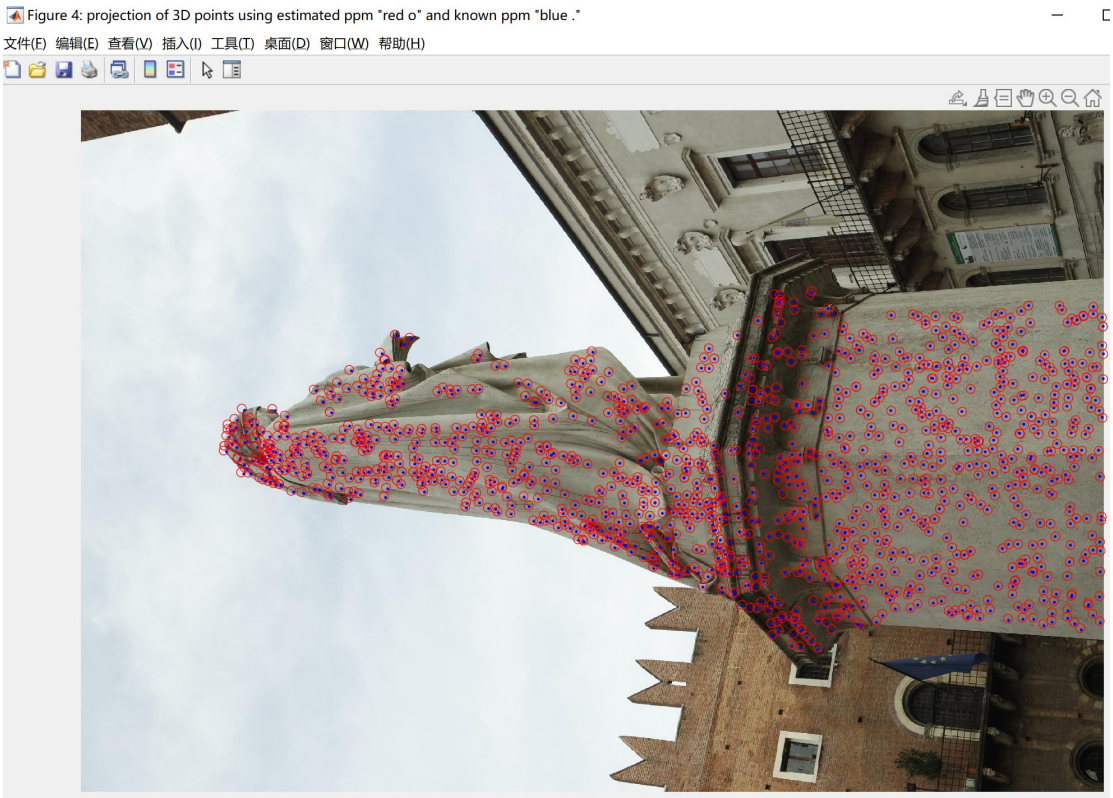
-0.1579	0.0831	0.9840	-72.1277
-0.3066	-0.9513	0.0311	61.2189
0.9386	-0.2968	0.1757	6.6508

estimated camera pose:

G_estimated =

-0.1599	0.0828	0.9837	-72.0699
-0.3046	-0.9520	0.0306	61.2794
0.9390	-0.2947	0.1774	6.4004

_SAM1002.JPG



known camera pose:

ans =

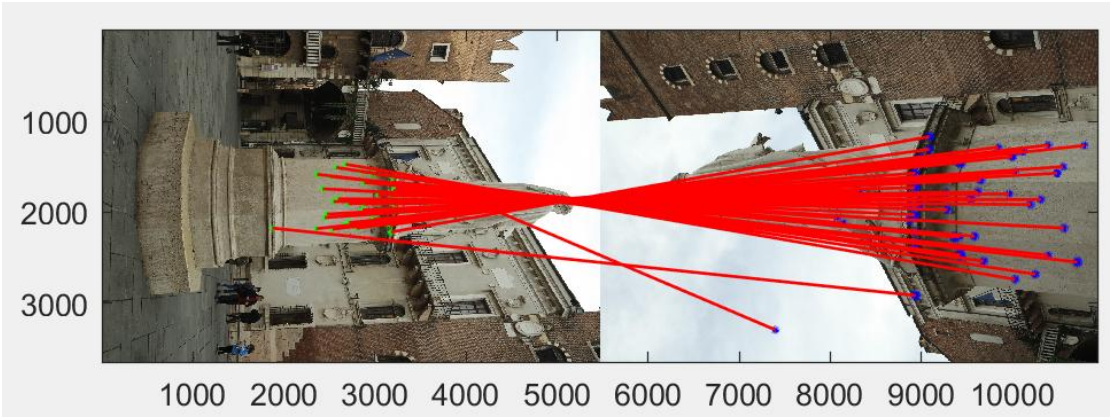
0.4922	0.0141	-0.8704	56.0879
-0.0736	0.9970	-0.0255	-60.8032
0.8674	0.0766	0.4917	-42.2190

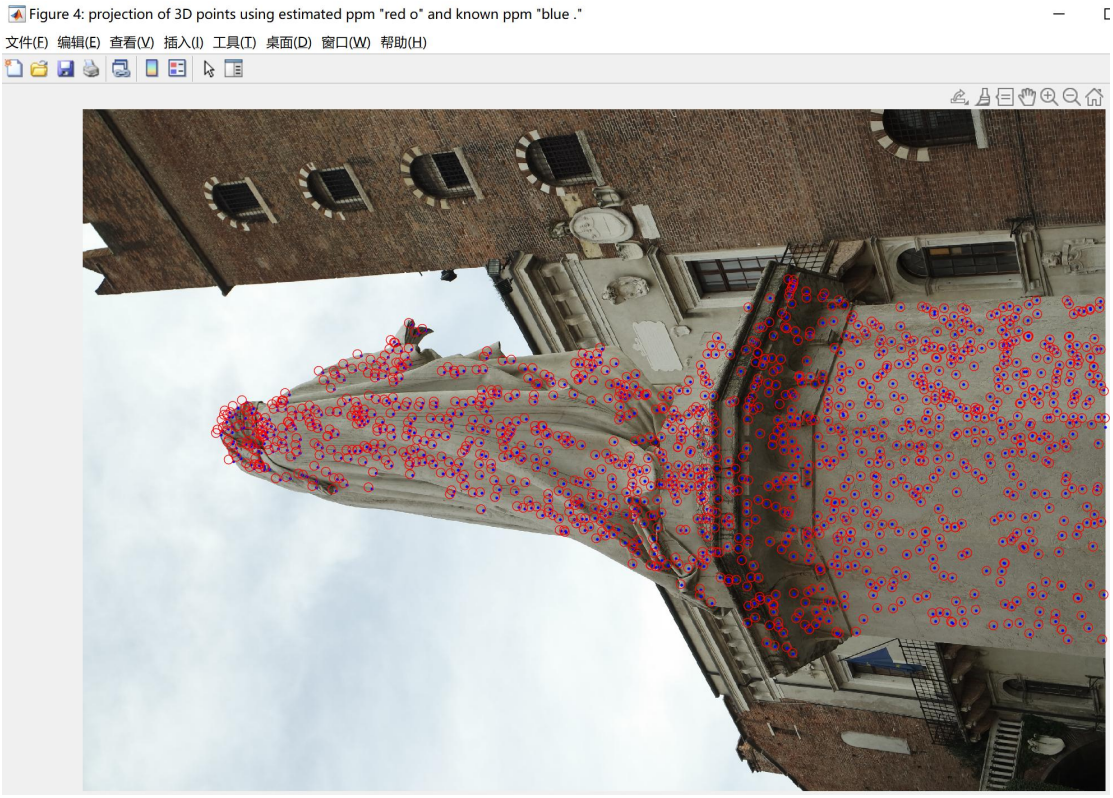
estimated camera pose:

G_estimated =

0.4853	0.0176	-0.8742	56.1944
-0.0836	0.9962	-0.0264	-60.5997
0.8703	0.0859	0.4849	-42.3515

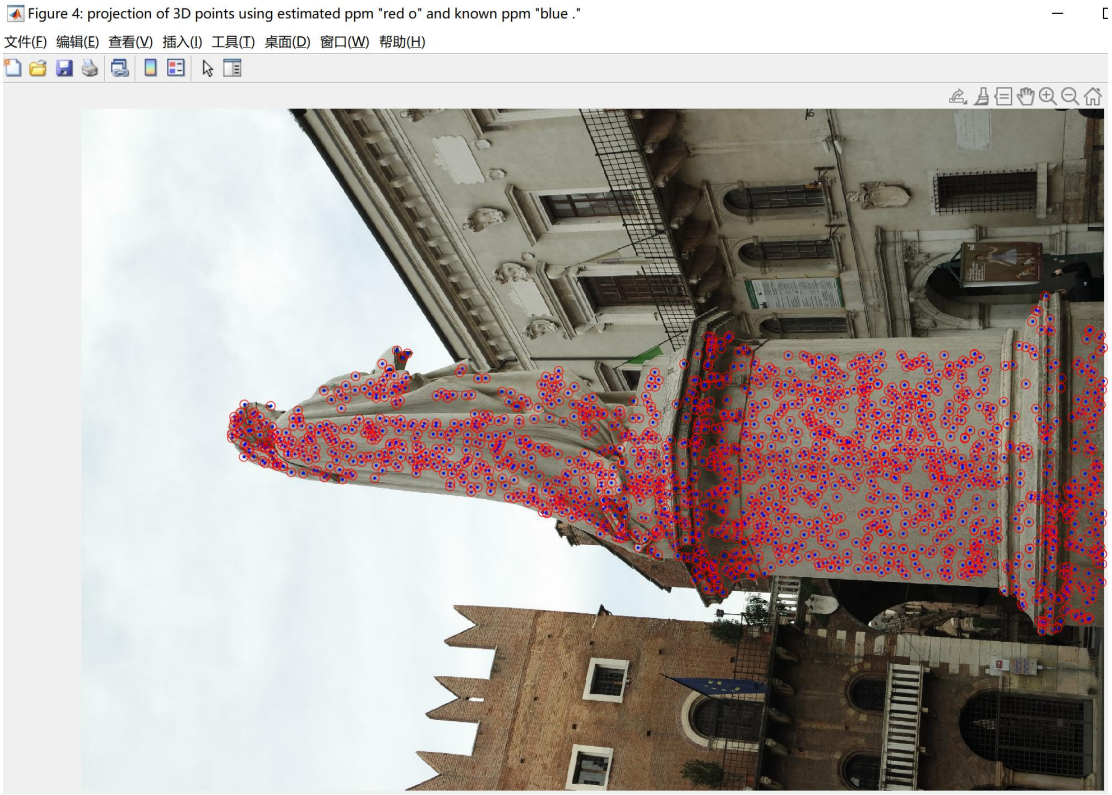
_SAM1079.JPG






```
known camera pose:

ans =

    0.4825   -0.1275   -0.8665   64.8652
    0.1719    0.9839   -0.0490  -60.6174
    0.8588   -0.1253    0.4967  -29.7142

estimated camera pose:

G_estimated =

    0.4734   -0.1242   -0.8720   65.1194
    0.1642    0.9851   -0.0512  -60.4777
    0.8654   -0.1190    0.4868  -29.4799
```

_SAM1080.JPG



known camera pose:

ans =

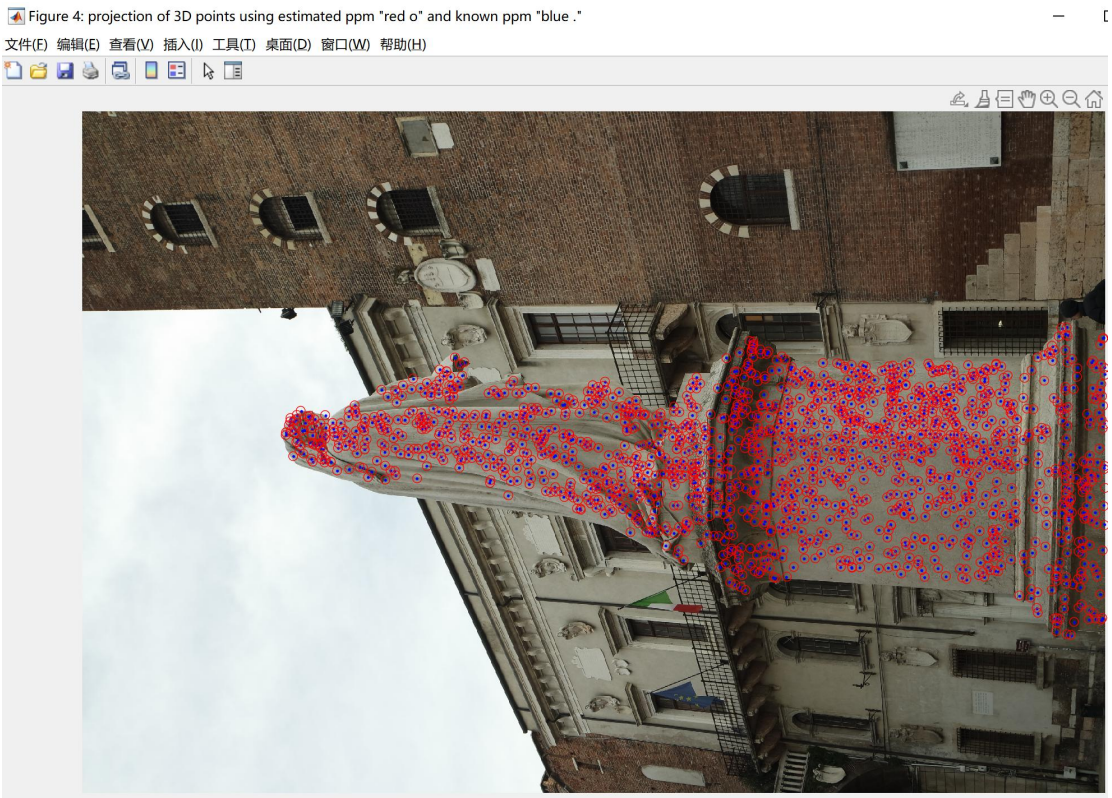
0.3290	-0.0343	-0.9437	65.6387
-0.0368	0.9981	-0.0491	-59.5786
0.9436	0.0508	0.3272	-27.8686

estimated camera pose:

G_estimated =

0.3253	-0.0331	-0.9450	65.6927
-0.0407	0.9980	-0.0490	-59.5449
0.9448	0.0543	0.3232	-27.8155

_SAM1107.JPG




```
known camera pose:

ans =

    0.3198   -0.1355   -0.9377   71.9663
    0.2218    0.9729   -0.0650  -59.2561
    0.9212   -0.1872    0.3412  -13.1343

estimated camera pose:

G_estimated =

    0.3160   -0.1317   -0.9396   71.8874
    0.2128    0.9749   -0.0651  -59.2961
    0.9246   -0.1794    0.3361  -13.2917
```

_SAM1108.JPG

Figure 12. *Projection of the 3D points using estimated ppm(red) and known ppm(blue), printed estimated camera pose and known camera pose*

4.2 Experiment 2 - Ca' Vignal 2 building

Input:

The image cav.jpg of the Ca' Vignal 2 building and its corresponding 3D points of the building were taken as the reference, as shown in Figure 13. As there is only one image provided by the professor, the same image cav.jpg also was taken as the target image.

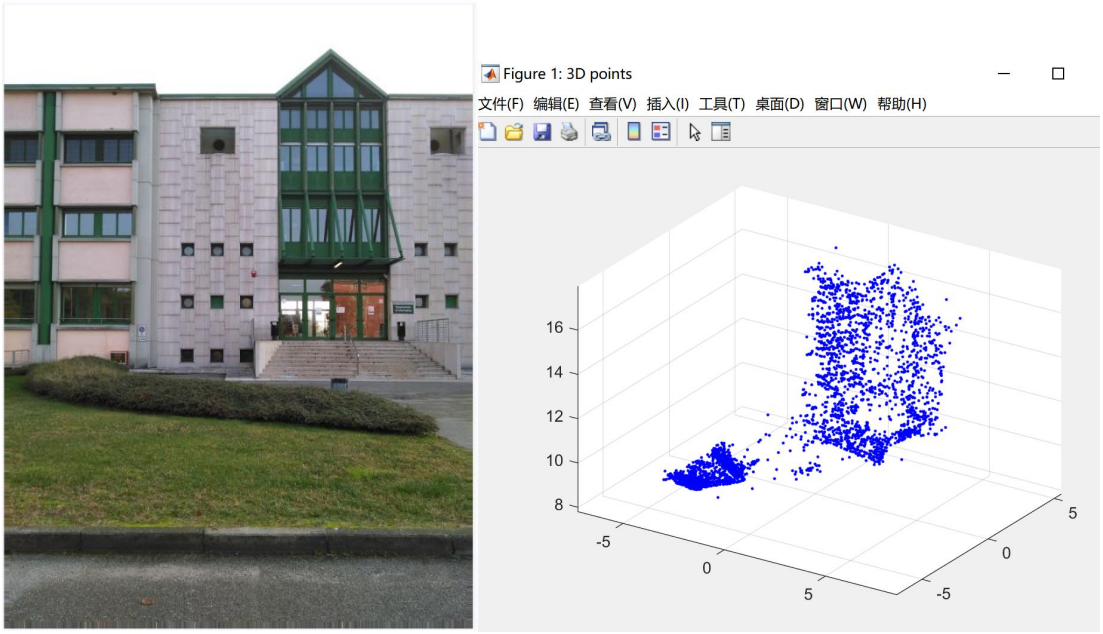
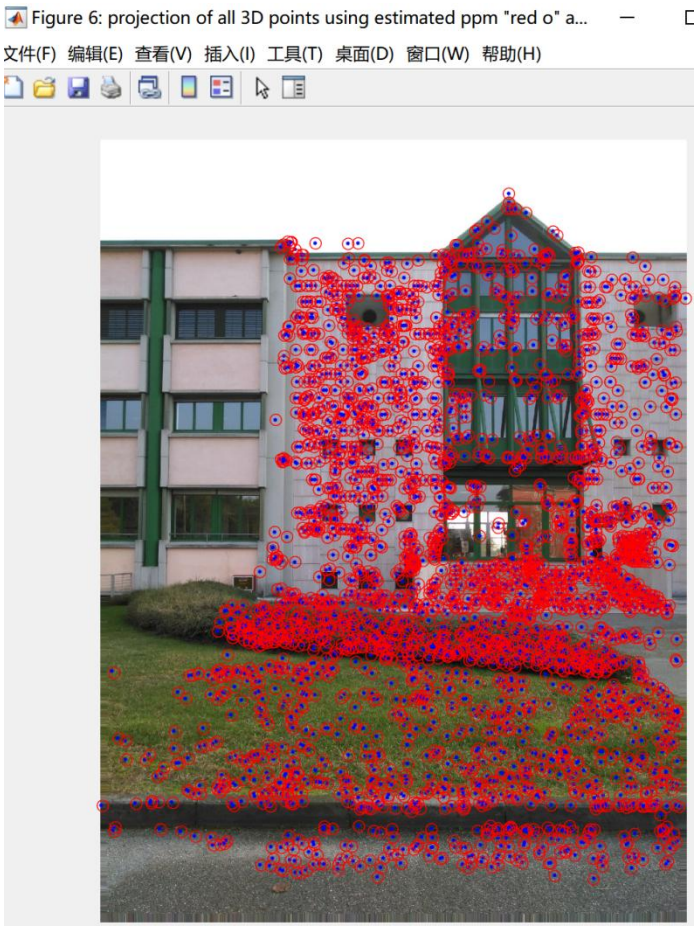


Figure 13. *Image* cav.jpg *and corresponding 3D points*

Result:

The estimated camera pose was tested by a comparison of the projections of the 3D points onto the target image using the estimated pose and the known one. The projections and the comparison of the estimated pose and the known pose are shown in Figure 14.




```
known camera pose:
ans =
    0.9871   -0.1603   -0.0012    1.7549
    0.0014    0.0161   -0.9999   10.7369
    0.1603    0.9869    0.0161    7.8101
estimated camera pose:
G_estimated =
    0.9871   -0.1601   -0.0010    1.7557
    0.0016    0.0158   -0.9999   10.7408
    0.1601    0.9870    0.0159    7.8192
```

Figure 14. *Projection of the 3D points using estimated ppm(red) and known ppm(blue), printed estimated camera pose and known camera pose*

5. Conclusions

The developed software application estimates the 3D -> 2D correspondences of a given scene and estimates the camera extrinsic parameters using Fiore's exterior orientation estimation method. Experimental validation shows that the algorithm performs well on camera pose estimation of images with different sizes.

6. Future Work

A possible further development of the project can be adjust and optimize the algorithm for applying it on a robot for real-time robot pose estimation. In this case, the camera is fixed on the robot and the algorithm takes the video stream as input for a real-time robot pose estimation.

7. References

[1]P. D. Fiore, "Efficient linear solution of exterior orientation," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 2, pp. 140-148, Feb. 2001, doi: 10.1109/34.908965.
[2] https://www.vlfeat.org/overview/sift.html#tut.sift.match
[3] https://www.vlfeat.org/api/sift.html
[4] M. A. Fischler, R. C. Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography, Readings in Computer Vision, 1987, Pages 726-740, https://doi.org/10.1016/B978-0-08-051581-6.50070-2.