Kin Hong Wong, Ying Kin Yu, Pang Kwan Tsui, Yin Fung Ho, and Ho Chuen Kam,"Robust and efficient pose tracking using perspective-four-point algorithm and Kalman filter", 2017 International Conference on Mechanical, System and Control Engineering (ICMSC 2017), St. Petersburg, Russia, during May 19-21, 2017.

Robust and Efficient Pose Tracking Using Perspective-Four-Point Algorithm and Kalman Filter

Kin Hong Wong, Ying Kin Yu*, Ho Yin Fung, Ho Chuen Kam
Department of Computer Science and Engineering line
The Chinese University of Hong Kong
*Hong Kong
e-mail: khwong@cse.cuhk.edu.hk, ykyu.hk@gmail.com

Kwun Pang Tsui
Department of Mechanical Engineering
The Chinese University of Hong Kong
Hong Kong
e-mail: warrentsui@outlook.com

***Abstract*—In this paper, we investigate the use of Kalman filter to enable robust tracking based on an efficient pose estimation algorithm, namely the four-point algorithm. Pose estimation is very useful in vision-based system control, for example in automatic driving and virtual reality inputs. Firstly, we have implemented a four-point pose estimation method with a personal computer. This estimation algorithm is supposed to be the method that requires the least number of point features for the generation of a unique solution. On the contrary, existing three-point algorithms may give multiple solutions. Then we have adopted a Kalman filter to enable robust tracking. Kalman filter is computationally efficient and very good at handling noise during tracking. The merge of these two techniques make us able to build a high-speed and yet robust system to be used in a wide variety of real applications. Furthermore, we have shown that a linear Kalman filter can be applied to filter off noises directly from the results of the four-point algorithm. Simulated and real data tests were performed and the results were satisfactory.**

Keywords-automatic control; pose estimation; virtual reality systems; kalman filter

I. INTRODUCTION

Computer vision-based pose estimation is useful in many industrial applications such as natural user inputs for virtual reality, machine assembly and automatic driving systems. In this paper, we investigate an efficient and robust method of pose estimation that uses only four feature points called four-point algorithm by Liu and Wong [1]. It belongs to the category of Perspective-n-Point approaches where n is equal to 4. The four-point algorithm is the most efficient method for pose estimation that requires the least number of features to produce unique results. In contrast, the three-point algorithm [2], which takes the minimum number of features for pose estimation may suffer from the problem of multiple solutions. Most four-point algorithms assume that the 3-D model is known and one image is enough to identify the object pose. During tracking, the object is moving and we need to track the object pose continuously. Kalman filter is then suggested in our project to achieve robust tracking. As for applications of such a method, for example in automatic driving, the model of a car in front of the camera is known. We can apply Kalman filter to track its pose. In this work, our attention is on how to use Kalman filter to enable robust object tracking. The pose of an individual frame can be acquired by the four-point pose estimation algorithm. We have found that the linear Kalman filter is suitable for post-processing to get an accurate pose sequence. It is the best choice since the solution is optimal [3]. More specifically, our system consists of two stages. Firstly, we need to have the model of the target object and the camera is calibrated. Then we track four feature points on the object using the CAMSHIFT algorithm in OPENCV [4]. Finally, the data are fed to the Kalman filter to enable robust tracking. From the experiments, we revealed that Kalman filtering is an effective strategy to reduce the noises in the tracking problem. The overview of the system is shown in Figure 1. The major contributions of this work are: (1) Our pose estimation approach is computationally efficient because the four-point algorithm involves no trigonometrically function. In theory, it can be implemented using simple lightweight computation machines or even Field Programmable Gate Array (FPGA) hardware. It has been applied to object tracking. (2) We show that Kalman filter can be used to enable robust pose tracking with the four-point algorithm. Unlike other pose tracking approaches that utilize extended Kalman filter, our solution is optimal under Gaussian assumption. The proposed system can acquire accurate pose sequences from the input image data.

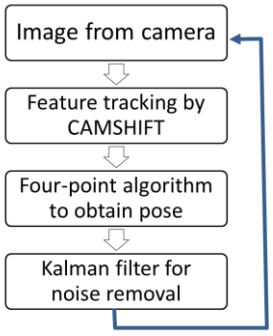

Figure 1. Overview of the four-point algorithm with Kalman filter.

This paper is divided into several parts: Section II explains the background of our project. In Section III, the theory and design methodology are discussed. The implementation and testing results are illustrated in Section IV. The conclusion is found in Section V.

II. BACKGROUND

Pose estimation is very useful in many engineering applications such as robotics, virtual reality and automatic part assembly. The four-point algorithm by Liu and Wong [1] can obtain the pose of an object using merely four feature points. Some research such as [5] and [6] extend this method and its application in building a 3-D sketch table can be found in [7]. A method to enhance its robustness is reported in [8]. Recently, the four-point algorithm has been applied to camera calibration in [9]. The Kalman filter is a recursive algorithm for state estimation in stochastic systems. It has been adopted for a numerous of applications since the 1960s. With the assumptions that the system is linear and both the process and measurement noise are Gaussian, the Kalman filter gives an optimal estimate of the system state in the sense of minimum-mean-square-error (MMSE), the maximum likelihood (ML), and the maximum a posteriori (MAP). The pose tracking problem that we are solving here consists of the linear dynamic and measurement model. The traditional form of linear Kalman filter is the most suitable choice to estimate the object pose in a sequential manner. Previously, Kalman filtering algorithms have been applied to recover both the 3-D model and pose information of a camera from an image sequence [10] [11]. The extended Kalman filter is used to deal with the non-linear measurement models in their work.

III. THEORY AND DESIGN

The overall setup is shown in Figure 2. The camera is tracking an object with four features of a known model. A CAMSHIFT algorithm is used to track these points and their 2-D image coordinates are fed to the four-point algorithm to find the pose. The pose consists of six parameters including translation (Tx, Ty, Tz) and rotation angles (θx, θy, θz) around the X, Y, Z coordinate axes. Kalman filtering is then applied to handle the pose tracking problem. Feature point tracking is handled by the CAMSHIFT algorithm in OPENCV [4]. The data are fed to the four-point algorithm for pose estimation as described below.

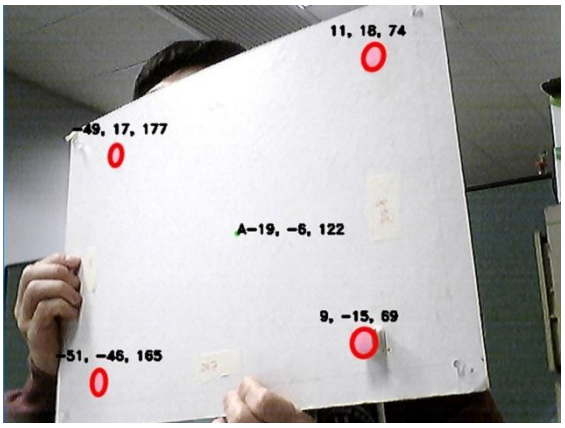

Figure 2. The A target object is being tracked. The point features obtained by the CAMSHIFT algorithm are shown in the picture. The 3-D (X,Y,Z) position of each feature calculated by the four-point algorithm is also shown.

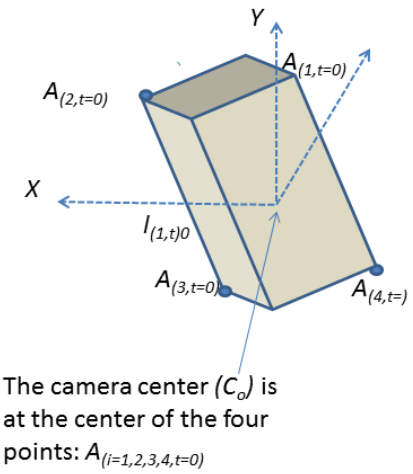

Figure 3. The coordinate system of the 3-D model.

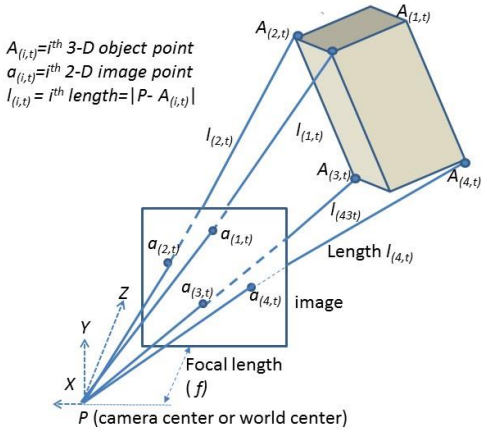

Figure 4. The perspective projection camera model

A. The Four-Point Algorithm

1) Problem definition:

Pose estimation is a method to determine the position and orientation of an object using a single image with a known object model. The four-point iterative pose estimation algorithm [1] that we use can compute the pose of a rigid body. Here is the description of this method. In Figures 3, and 4, the object consists of four model points. When the object center is the same as the camera center, the distances between these points to the camera center form the model.

From the model, the relative distances $D(i,j) = |A_i - A_j|$ among these points should be found. At time t, these feature points are extracted and tracked by an algorithm like CAMSHIFT [4]. They are denoted by $a_{1,t}, a_{2,t}, a_{3,t}, a_{4,t}$. With these image measurements and the focal length f of the camera, we can compute $l_{1,t}, l_{2,t}, l_{3,t}, l_{4,t}$. The four lengths can be converted to the rotation and translation of the object easily by simple geometric calculations. In Figure 4, the image points are denoted by $a_{n,t}$ and the 3-D points are $A_{n,t}$, where n = 1,2,3,4. The four lengths can be converted to the

rotation and translation of the object easily by simple geometric calculation. In Figure 4, the image points are denoted by $a_{n,t}$ and the 3-D points are $A_{n,t}$, where n = 1,2,3,4. The time index t is skipped for the sake of simplicity, since the pose estimation process can be completed in one time step. We see that a vector passing through the camera center P and the 3-D point A is $\overrightarrow{PA_n} = l_n\overrightarrow{u_n}$.

Using the cosine rule described in [12], we can form the following relation as follows.

$$D_{i,j}^2 = \left(A_i - A_j\right)^2 = l_i^2 + l_j^2 - 2l_i^2 l_j^2\left(\overrightarrow{u_i} \cdot \overrightarrow{u_j}\right) \quad (1)$$

The target is to find the length l_n of the vector $\overrightarrow{PA_n}$. To achieve this goal, we require the image measurements a_n and focal length f, where n = 1,2,3,4. The unit vector of $\overrightarrow{PA_n}$ or $\overrightarrow{Pa_n}$ is $\overrightarrow{u_n}$. Hence, the distance between Ai and Aj is $D_{i,j}^2 = l_i^2 + l_j^2 - 2l_i^2 l_j^2\left(\overrightarrow{u_i} \cdot \overrightarrow{u_j}\right)$. Assuming that the lengths predicted are l_n', the difference of the prediction and the true one is

$$e(i,j) = D_{i,j}^2 - \{l_i'^2 + l_j'^2 - 2l_i' l_j'\left(\overrightarrow{u_i} \cdot \overrightarrow{u_j}\right)\} \quad (2)$$

By arbitrarily choosing six different combination of i and j, we can obtain the following combinations to form these error terms: $e_a = e_{(i=1,j=2)}, e_b = e_{(i=1,j=3)}, e_c = e_{(i=1,j=4)}, e_d = e_{(i=2,j=3)}, e_e = e_{(i=2,j=4)}, e_f = e_{(i=3,j=4)}$. As discussed in [1], multiple solutions may occur if only these six constraints are used. To solve the problem, their paper introduces an additional rule. As shown in Figure 5, the projection of the vector $\vec{V}$ on $\overrightarrow{A_{3,2}}$ is the scalar value r, hence, $r = \{(\overrightarrow{A_{4,3}} \times \overrightarrow{A_{3,1}}) \cdot (\overrightarrow{A_{3,2}})\}$ is based on the given model with known $A_{1,t=0}, A_{2,t=0}, A_{3,t=0}, A_{4,t=0}$.

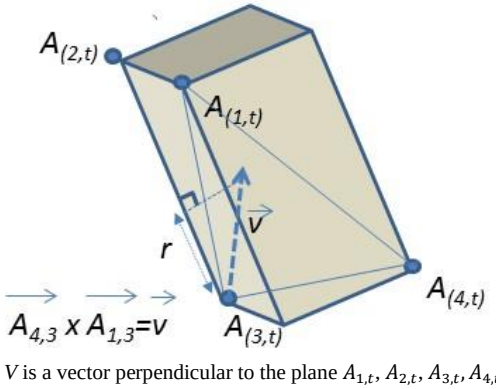

Figure 5. V is a vector perpendicular to the plane $A_{1,t}$, $A_{2,t}$, $A_{3,t}$, $A_{4,t}$, so r is a constant irrespective of the pose of the object.

This projection scalar r is a constant irrespective to pose and time t. If the guessed lengths are l_i', we can find the corresponding projection r' as shown below.

$$r' = \{[(l_3'\overrightarrow{u_3} - l_4'\overrightarrow{u_4}) \times (l_1'\overrightarrow{u_1} - l_3'\overrightarrow{u_3})] \cdot (l_2'\overrightarrow{u_2} - l_3'\overrightarrow{u_3})\}$$

Therefore, a new error term becomes.

$$e_g = r - r' \quad (3)$$

$$e_g = \left\{(\overrightarrow{A_{4,3}} \times \overrightarrow{A_{3,1}}) \cdot \overrightarrow{(A_{3,2})}\right\} - \{[(l_3'\overrightarrow{u_3} - l_4'\overrightarrow{u_4}) \times (l_1'\overrightarrow{u_1} - l_3'\overrightarrow{u_3})] \cdot (l_2'\overrightarrow{u_2} - l_3'\overrightarrow{u_3})\}$$

Then seven constraints are established. Using $e_a, e_b, e_c, e_d, e_f, e_g$ and equation 3, we stack up all e's to form an error vector $E = \left[e_a, e_b, e_c, e_d, e_f, e_g\right]^T$.

2) Iterative processing

With error vector $E = \left[e_a, e_b, e_c, e_d, e_f, e_g\right]^T$, Jacobian matrix $J = \frac{\partial E}{\partial l}$ and Δl, we can use the Gauss-Newton method to iteratively minimize the error. At iteration k, which is from 0 to maximum allowable k, we have

$$L^{k+1} = L^k + \Delta l$$

$$J(\Delta l) = E$$

and

$$J(\Delta l) = E$$

This will find all lengths $l_{(n=1,2,3,4)}$ and all $A_{(n=1,2,3,4)}$ at time t. Unlike other pose estimation approaches such as [10], there is no need to use any trigonometric function during execution. So potentially it can be implemented using very simple hardware. Since it only uses four feature points, the computation complexity is low. An extension of this approach can use RANSAC scheme [13] to select groups of points to make the calculation more robust.

B. Kalman Filter

1) Inputs to the Kalman filter

Since the object model of the four points is supposed to be known at time t = 0, we can assume that the center of the four points

$A_{center,t=0} = mean\left(A_{1,t=0}, A_{2,t=0}, A_{3,t=0}, A_{4,t=0}\right)$. At time t, the object is moved to a new position defined by rotation R_t and translation T_t. After the four-point algorithm is executed, the four 3-D points $\left(A_{1,t=t}, A_{2,t=t}, A_{3,t=t}, A_{4,t=t}\right)$ are found. It is then straight forward to find the pose of the object. We assume that a rotation matrix R_t rotates all features from time t to $t+1$ around the rotating center at $A_{center,t}$, which is the center of the object feature points at time t. Only 3 points are needed in the computation. If all 4 points are utilized, one can use pseudo inverse to obtain the rotation R_t. Therefore, we can find R_t if (A_t, A_{t+1}) are found. With $T_t = A_{center,t}$, R_t and T_t are converted to a state vector of 6 elements $z_t = \left[T_x, T_y, T_z, \theta_x, \theta_y, \theta_z\right]_t^T$, which is the measurement input for the Kalman filter.

2) Iterative Kalman filter algorithm

We adopted the Matlab Kalman toolbox in [14] for our experiment. The detailed formulation of the Kalman filter can be found in user manual of the toolbox. We employed standard system and measurement noise setting for our tests. The effects of tuning these parameters will be investigated in future. The theory is described below. The basic transfer functions are: $x_t = Ax_{t-1} + w_t$, and $z_t = Hx_t + v_t$. From the [3], $x_t = \left[T_x, T_y, T_z, \theta_x, \theta_y, \theta_z, \dot{T}_x, \dot{T}_y, \dot{T}_z, \dot{\theta}_x, \dot{\theta}_y, \dot{\theta}_z\right]_t^T$, is the state vector, t is the time index and z is the measurement. The process transfer function is Matrix A, which describes the dynamics of the system process. Matrix H describes the observation model. It is an identity matrix in our case. P is

the error covariance matrix. *w* represents the process noise with covariance *Q*. *v* is measurement noise with covariance *R*. The Kalman filter consists of the prediction and update step. One can refer to the other literature for the details. After initialization, the two steps are executed for all measurements up to z_t to generate the stabilized state vector x_t.

Algorithm 1 The Four Point Pose Estimation Algorithm

Input: a_1, a_2, a_3, a_4 and model of the object $A_{(i=1,t=0)}, A_{(i=2,t=0)}, A_{(i=3,t=0)}, A_{(i=4,t=0)}$, focal length=5 micrometers

Output: $L = [l_1, l_2, l_3, l_4]^T$

Initialize: $k = 0$, approximated guess of l'_1, l'_2, l'_3, l'_4.

while $k < maximum\_K$ or $\|E\| <$ threshold **do**

$E^{(k)} = [e_a, e_b, e_c, e_d, e_e, e_f, e_g]^T$

$$\Delta l = \begin{bmatrix} l_1'^{(k+1)} - l_1'^{(k)} \\ l_2'^{(k+1)} - l_2'^{(k1)} \\ l_3'^{(k+1)} - l_3'^{(k)} \\ l_4'^{(k+1)} - l_4'^{(k)} \end{bmatrix}$$

$\Delta l' = ((J^T J)^{-1} J^T) E^{(k)}$

$E^{(k+1)} = J \Delta l'$

$L'^{(k)} = [l_1'^{(k)}, l_2'^{(k)}, l_3'^{(k)}, l_4'^{(k)}]^T$

$L'^{(k+1)} \leftarrow (L'^{(k)} + \Delta l')$

$k \leftarrow k + 1$

end while

$L \leftarrow L'^{(k)}$

IV. Experiments and Results

In the simulation experiment, we would like to see how the four point algorithm and Kalman filter are affected by motion fluctuations. A simulation result is shown in Figures 6 and 7. The horizontal axes of these plots are the time steps. These 2 figures help us explain our experimental procedures. (1) We created a 3-D model of four points similar to that shown in Figure 2. (2) We generated a ground truth motion path *gt(t)* with certain variations for the object being tracked, which are shown as the (+) lines in the figures. (3) Motion fluctuation (for each of the six parameters of the pose) was controlled by a noise factor η of standard deviation σ_n. The observed motion *obs(t)* is shown as the (□) lines. (4) This observed motion was used to generate the four image feature positions. A 1-pixel standard deviation random noise was added to the feature image positions to simulate the noise encountered during the feature tracking process. (5) The image features were processed by the four-point algorithm to obtain the pose sequence *4p(t)* depicted as the (*) lines. This pose sequence *4p(t)* was sent as the input for the Kalman filter. (6) The outputs of the Kalman filter *ko(t)* are shown as the (O) lines in the figures. We see that the Kalman filter output *ko(t)* is closer to the ground truth motion path *gt(t)* than that of the observed path *obs(t)* i.e. the input to the system. This shows that the Kalman filter can filter some of the motion fluctuations. To see how motion fluctuation is related to the Kalman output, we performed the following procedures. For each η value, we carried out 100 tests and each is indexed as *j*. In the j^{th} test, we ran the above steps from 2 to 6 to obtain $ko(j,t,\eta)$. We then calculated the *std()*standard deviation $\sigma_{(gt-ko),\theta,j,\eta} = \{std((gt(t) - ko(t,\eta))\theta_x + std((gt(t) - ko(t,\eta))\theta_y + std((gt(t) - ko(t,\eta))\theta_z\}/3$ for the rotation. We also computed $\bar{\sigma}_{(gt-ko),\theta,\eta} = mean(\sigma_{a,\theta,j})$ for all *j*. Similarly, we found $\bar{\sigma}_{(gt-ko),T,\eta} = mean(\sigma_{(gt-ko),T,j})$ for the translations. Based on the this scheme, we also calculated $\bar{\sigma}_{(gt-obs),\theta,\eta}$ and $\bar{\sigma}_{(gt-obs),T,\eta}$ between *gt(t)* and *obs(t)*. We plotted these $\bar{\sigma}$ values against the increasing η in Figure 8. We found that $\bar{\sigma}_{(gt-obs),T}(\diamondsuit)$ and $\bar{\sigma}_{(gt-obs),\theta}$ $(\diamondsuit)$ are higher than $\bar{\sigma}_{(gt-ko),T}(\Delta)$ and $\bar{\sigma}_{(gt-obs),\theta}(\Delta)$, respectively, as η increases. It indicates that the Kalman filtered output *ko* is closer to the ground truth *gt* motion that between the *gt* and *obs* (*obs* is the input to the Kalman filter). The Kalman filter is able to stabilize the system. More information of this work can be found at http://www.cse.cuhk.edu.hk/_khwong/papers.html.

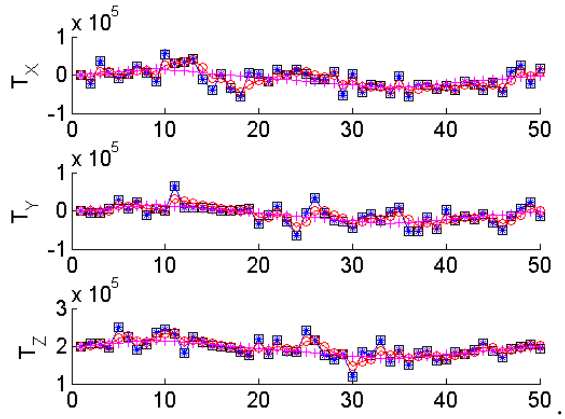

Figure 6. A sample of the simulation results for translations (T_x, T_y, T_z) in pixles verus *time* step *t*: observed *obs(t)*=(□), Kalman input *4pt(t)*=(*), Kalman output ko=(O),ground truth *gt(t)*=(+).

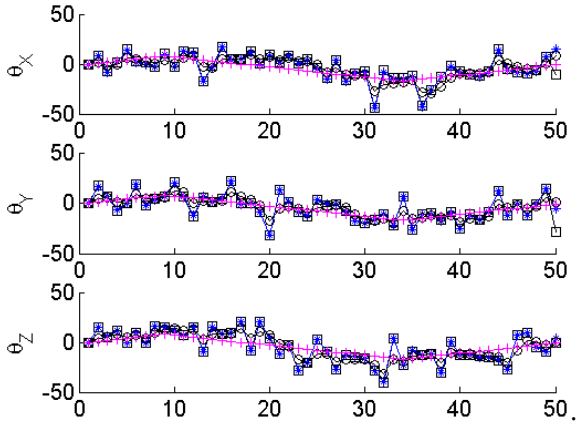

Figure 7. A sample of the simulation results for rotation angles ($\theta_x, \theta_y, \theta_z$) in degrees verus *time* step *t*: observed *obs(t)*=(□), Kalman input *4pt(t)*=(*), Kalman output ko=(O),ground truth *gt(t)*=(+).

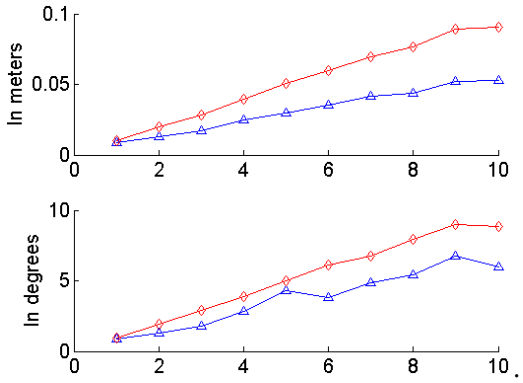

Figure 8. Top: $\bar{\sigma}_{(gt-obs),T} = \diamondsuit$, $\bar{\sigma}_{(gt-ko),T} = \Delta$ for translation verus η on the horizontal axis. Each η unit represents 0.01 meters. Bottom:$\bar{\sigma}_{(gt-obs),\theta} = \diamondsuit$, $\bar{\sigma}_{(gt-ko),\theta} = \triangle$ for rotation angles verus η on the horizontal axis.Each η unit represents 1 degree. It shows that the Kalman filtered output is able to stabilize the system. Note: *gt*=ground truth, *obs*=observation=*input to the Kalman filter*, *ko=Kalman filter output*,*T=Translation, θ=rotation angle*

V. CONCLUSION

In this work, we have applied Kalman filter to increase the robustness of a pose estimation algorithm called the four-point algorithm. The four-point algorithm is an effective method for determining the pose of an object using only four model points. It is based on one image and the resulting solution is unique. We take the advantages of the dynamic system in the Kalman filtering framework to achieve robust tracking of the object motion. The proposed method is useful in many engineering applications such as vision-based automatic driving. In the experiment, we see that the Kalman filtering procedure is able to reduce noises in the system and enable robust tracking of the object pose.

ACKNOWLEDGMENT

This work is supported by a direct grant (Project Code: 4055045) from the Faculty of Engineering of the Chinese University of Hong Kong.

REFERENCES

[1] Man Lee Liu and Kin Hong Wong. Pose estimation using four corresponding points. Pattern Recognition Letters, 20(1):69–74, 1999.

[2] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24(6):381–395, 1981.

[3] Yaakov Bar-Shalom, X Rong Li, and Thiagalingam Kirubarajan. Estimation with applications to tracking and navigation: theory algorithms and software. John Wiley & Sons, 2004.

[4] Itseez. (2015). opencv. http://opencv.org/. Accessed: 2016-03-04.

[5] Pengfei Sun, Changku Sun, Wenqiang Li, and Peng Wang. A new pose estimation algorithm using a perspective-ray-based scaled orthographic projection with iteration. PloS one, 10(7):e0134029, 2015.

[6] Zimiao Zhang, Shihai Zhang, and Qiu Li. Robust and accurate vision based pose estimation algorithm based on four coplanar feature points. Sensors, 16(12):2173, 2016.

[7] Alfredo Liverani, Alessandro Ceruti, and Gianni Caligiana. Tablet based 3d sketching and curve reverse modelling. Int. Journ. of Comp. Aided Engineering and Technology 8, 5(2-3):188–215, 2013.

[8] Zimiao Zhang, Bin Liu, and Yongxiang Jiang. A two-step pose estimation method based on four non-coplanar points. Optik-International Journal for Light and Electron Optics, 126(17):1520–1526, 2015.

[9] Zhe Zhang and Kin Hong Wong. A novel geometric approach for camera calibration. In Image Processing (ICIP), 2014 IEEE International Conference on, pages 5806–5810. IEEE, 2014.

[10] Ying Kin Yu, Kin Hong Wong, and Michael Ming-Yuen Chang. Recursive three-dimensional model reconstruction based on kalman filtering. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 35(3):587–592, 2005.

[11] [11] Ying Kin Yu, Kin Hong Wong, Michael Ming-Yuen Chang, and Or Siu Hang. Recursive camera-motion estimation with the trifocal tensor. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 36(5):1081–1090, 2006.

[12] John Bird. Engineering mathematics. Routledge, 2003.

[13] Konstantinos G Derpanis. Overview of the ransac algorithm. Image Rochester NY, 4(1):2–3, 2010.

[14] Jouni Hartikainen, Arno Solin, and Simo S ̈arkk ̈a. Optimal filtering with kalman filters and smoothers. Department of Biomedica Engineering and Computational Sciences, Aalto University School of Science, 16th August, 2011.