3D Pose Estimation and Tracking of an Electricity Pylon

Emmanuel Y. Ali
Electrical Engineering
University of Cape Town
alxemm004@myuct.ac.za

Fred Nicolls
Electrical Engineering
University of Cape Town
fred.nicolls@uct.ac.za

***Abstract*—We demonstrate an algorithm for estimating the 6 degree-of-freedom pose of a textureless object such as a power pylon. The method is designed to be part of the inspection process for a transmission line. The approach involves three steps, namely predicting the vertices of the pylon with a neural network, establishing the correspondence between vertices with hashing techniques, and incrementally tracking the movement of the camera. We built a pylon model for the experiments and used it to generate our dataset.**

***Index Terms*—3D Object detection, Pose estimation, Transmission line inspection, Tracking**

I. Introduction

The transmission line is a vital part of our lives as it is used to supply electricity. Therefore, we need to ensure it is always maintained and that each component is functioning properly. One part of the inspection process involves identifying individual power pylons and making sure that none of the parts are missing or malfunctioning. Foot patrols and helicopters are often used to do the inspection. Robots are starting to be used for inspection, although they are mostly remotely controlled.

The first task needed for automating the inspection process is to accurately localise the objects of interest. A lot of the literature on visual inspection focuses on object detection with the output as a bounding box, such as pylon detection [1]–[3] and insulator detection [4], [5]. Localisation or pose estimation involves finding the six degree of freedom position and orientation of an object relative to the camera. The pylon is the most prominent component of the transmission line and localising it provides a context for locating other components that need to be inspected.

The pylon is made up of bars or struts intersecting at points or vertices. It belongs to the category of textureless "wiry objects" for which most stereo reconstruction or tracking methods fail — an image patch around a feature point relates to the background and not the object itself, so pixel descriptors cannot be used for matching. SLAM-based methods also fail in this situation.

To develop the ideas presented in this work we built a model pylon for use in the experiments. The dataset of the model pylon was generated from images taken at various angles and against different backgrounds. Figure 1 shows an example image with some results of the proposed algorithms indicated. The model was built from plan by hand using architectural model material. Insofar as the building is accurate the 3D locations of vertices and struts in the object frame are known.

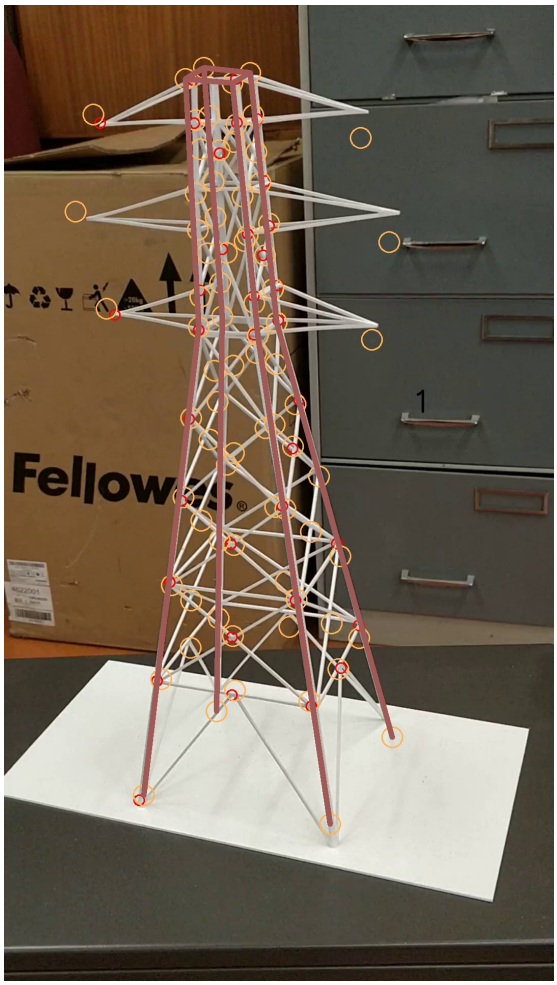

Fig. 1: An example of result of the algorithm with the estimated 3D model of the pylon projected onto the image. The vertex prediction of the CNN is shown in red.

The work addresses the localisation problem of estimating the 6 degree-of-freedom pose of a camera relative to the pylon.

978-1-7281-4162-6/20/$31.00 ©2020 IEEE

To achieve this we use a convolutional neural network with a deep residual network (Resnet) archictecture as a keypoint detector for a vertex, and geometric hashing to establish correspondence between vertices. Once an initial pose estimate is available an extended Kalman filter is proposed for tracking incremental changes in the pose of the camera.

The structure of this paper is as follows. In section II we talk about related work on pose estimation. The means of acquiring images plays a vital role in the technique. Section III describes the algorithm framework. This includes object representation and image features that are easily extracted. We discuss the search strategy and how it can register the 3D model onto the image. Incrementally tracking the movement of the camera is also described. In section IV we provide details of the implementation and the results obtained in the research. We summarise our research findings in section V.

II. Literature review

Pose estimation is a classic computer vision problem and different techniques have been used to solve the problem. Feature-based techniques such as SIFT [6], SURF [7] and ORB [8] work well with natural images that are well textured. These methods involve extracting keypoint features from the images. Using a descriptor method, they recognise similar characteristics across scale, orientation and location variations in order to match between images.

RGB-D cameras provide depth information and have been used to solve pose estimation for weakly textured objects. A template matching technique was used by [9] for the 3D pose estimation problem, where they used the color and depth information to establish a match with a template from different views of the object. In [10] a generalized Hough-like voting technique for pose estimation is used, where they extracted the features using oriented point pair features. The use of random forests was proposed by [11]. The random forest was used to predict 3D object coordinates and instance probabilities of the object. The output of the network was used in a cost function, and a RANSAC based method provided the pose estimate.

Work that is closely related to our research is [12]. They consider pose estimation and tracking for augmented reality (AR). A CNN is used for the part-based detection. Each part is a discriminative region of the object and represents a 2D reprojection of the 3D points. Their 3D points do not represent any special feature but are arbitrary points used to provide information for achieving a 2D-3D projection. The network output is invariant to the image location on the parts and depends on the image appearance. The network detects several candidates for the parts and the most likely candidates given the prior on the pose, represented as a mixture of Gaussians, are selected. The prior is used to define the normal range of the camera. The pose computed for previous frames is incorporated into the system for temporal consistency. Optimization using the Gauss-Newton algorithm is done over the prior and the parts to solve for the pose estimates. For the best pose estimate they evaluate the pose under different cues, and train a linear regressor to predict the penalty for the different cues. An extended Kalman filter smoothes out the trajectory and reduces jitter.

III. Algorithm framework

We need to obtain the pose estimate of the pylon and also track the movement of camera relative to the pylon. The pylon is a repetitive structure and is not well-textured, making it a difficult problem. To handle the problem we need a way to represent the object. The vertex is a unique point on the pylon and a collection of vertices serves as the representation of the object. We register the 3D model of the pylon onto the image, allowing us to obtain the 6 degree of freedom (DOF) pose estimate of the pylon relative to the camera. Then we track the movement of the camera around the pylon.

In the sections below we discuss vertex detection using a convolutional neural network, registration of a view of the 3D model of the pylon using geometric hashing, and tracking the movement of the camera using an extended Kalman filter (EKF).

A. Keypoint based detection of a vertex

The use of convolutional neural networks has become very common in a lot of computer vision tasks. With the concept of transfer learning, a network architecture can be used for different tasks. There are different CNN models that have been pretrained on large datasets like ImageNet. The network architecture we use was borrowed from the deepercut [13] paper on human pose estimation and the deeplabcut [14] paper for hand annotation.

We use the ResNet [15] pretrained architecture for this work. The ResNet configuration was designed for deep neural networks. It was observed that as the depth of the neural network increases the feature map output degrades, and ResNet was one method developed to address this issue. Most networks are made for classification but can be fine tuned for other tasks such as object detection, segmentation and keypoint detection. ResNet is made up of five different blocks comprising convolutional filters, pooling filters and non-linear filters in each block, and a final layer of average pooling and classification.

For the detection of the vertex, which is keypoint detection, we need to modify the ResNet to fit our task. The final layer of the average pooling and the classifier is removed. The removal of the final layer means the feature map has an output stride of 32 pixels. It tends to be coarse for localization. Therefore, the 3×3 convolutional filter in the fifth block is replaced by a dilated convolutional filter using the hole algorithm: the dilated convolutional filter puts zeroes (holes) in between the kernel elements. Finally, the output is passed through a fractional convolution filter of stride 2 pixels to upsample the feature map to an output stride of 8 pixels.

1) Training: There are different ResNets with differing depths, and for this experiment we used the ResNet 50.

Training is done with a sigmoid activation and a cross-entropy loss function. Stochastic gradient descent is used as the optimizer. The dataset is divided into a ratio of 75% for

the training and 25% for the testing stage. To train the model we need to create a heat map. The heat map has the size of the resized image and a depth equivalent to the number of vertices. Each of the channel of the heatmap where a particular vertex is located is labeled as 1 otherwise the rest of is labeled as 0.

The pylon is symmetric in that opposite faces are identical, and both can be seen simultaneously from each side. In the models for the training we mostly focused on the front face.

For the training of the CNN we collected images from different positions and orientations. We used varying backgrounds for the data collected to ensure variety in the images. We labeled each visible vertex point in each image uniquely, with a maximum of 77 vertices for each view of the pylon.

The network used 1.03M iterations for the training, which is equivalent to 3433 epochs. The learning rate starts from 0.001 for the first 10k iterations, then uses 0.002 for the next 420k, 0.0002 for the next 300k, and 0.0001 for the next 300k.

2) Results: We used 300 images of the pylon for the task of training and testing the vertex detection network.

A maximum pixel location in each channel of the predicted heatmap is considered as the vertex output. This provides the vertices which are keypoints and used for the geometric hashing and with the Kalman Filter.

The end result was a training error of 22.81 pixels and an error in the testing stage of 42.48 pixels.

Images	Image 1	Image 2	Image 3	Image 4	Image 5
Number of points	32	53	32	32	32
Correct prediction	30	48	15	28	30

TABLE I: Table for the correct prediction of the vertex.

Table I provides information on the effectiveness of the CNN for predicting the vertex positions for certain selected images. The "Number of points" is the total number of labeled points that we hope to find. In many cases these points are on the side of the pylon facing the camera. The figures quoted under "Correct prediction" contain counts of the number of times the network correctly identifies the vertices.

The prediction is good, as can be seen in the table and in Figure 2, which shows both labelled vertices and their predictions given by the network.

B. 3D pose detection

Geometric hashing [16] is a model-based technique that we use for the pose detection problem. It is implemented in two stages, the training and the recognition stage. The training stage involves representing and storing the model and its features in a hash table. The index of the hash table is the weighted value of a point in the reference frame and the model, and a subset of the features are its accompanying information in the hash table. The subset of the features is an ordered pair called the basis set and the number of features to form a basis set is determined using the transformation type.

The basis set is used to transform a point to a new reference frame weighted value. We see in Equation (1) the relationship between the basis set, the basis set center p_c, and a point p_i:

$$p_i - p_c = ap_x + bp_y. \tag{1}$$

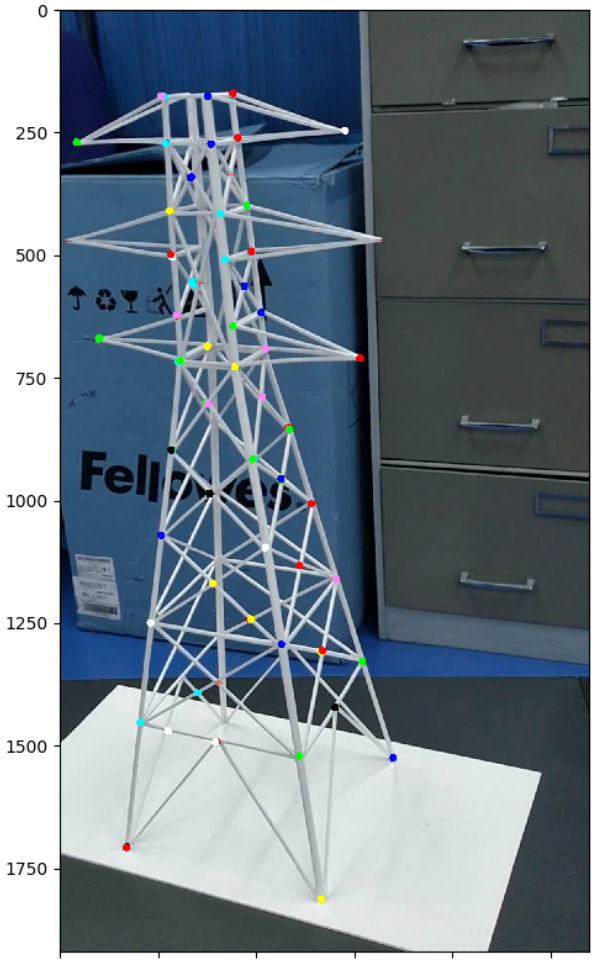

Fig. 2: An image of the the CNN hand annotated label (+) and the predicted output of the network (•).

We solve for the index (a, b) by rearranging (1) to give the formulation as

$$Ax = y \tag{2}$$

with the basis set matrix

$$A = \begin{bmatrix} p_{x1} & p_{y1} \\ p_{x2} & p_{y2} \end{bmatrix}, \quad y = \begin{bmatrix} (p_{i1} - p_{c1}) \\ (p_{i2} - p_{c2}) \end{bmatrix} \quad \text{and} \quad x = \begin{bmatrix} a \\ b \end{bmatrix}. \tag{3}$$

The sides of the pylon are planar, so any two views of a side are linked by a planar homography or projective transformation. In our work we use an affine transformation, which should be approximately valid as long as the camera is quite far from the pylon. We require three points to determine the affine transformation. These points can be the vertices of the triangle (p_0, p_1, p_2) to represent the basis set, and the center of the basis set is $p_c = (p_0 + p_1 + p_2)/3$.

The basis set matrix columns are the difference between the two points against one of the points $p_x = p_1 - p_0$ and $p_y = p_2 - p_0$.

To form the hash table we pick any three points in Figure 3 as the basis set, and use equation (1) to transform the other

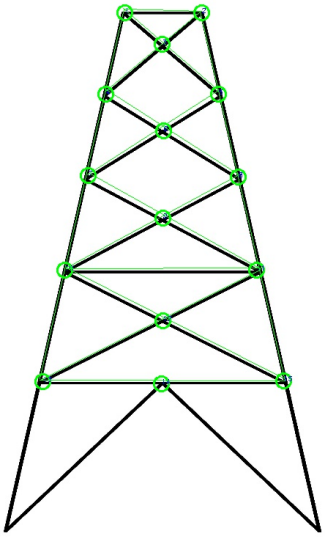

Fig. 3: Training image.

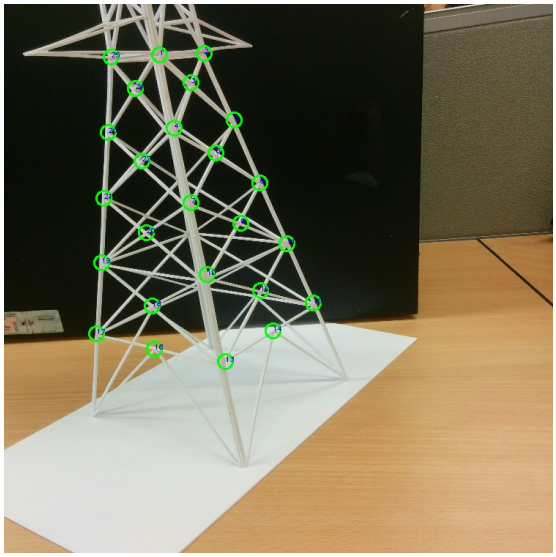

Fig. 4: Recognition image.

points to a new reference frame ($p_i \rightarrow (a, b)$). The new position of the points is weighted and quantized to represent the index (a_s, b_s) for the hash table. This is done for every possible combination of three non-collinear points.

The recognition stage involves having a scene image and matching it to a model in the training stage. We use the basis set to transform all the other features. A voting technique attempts to align the scene features with the training features. If this happens then the basis set with the highest vote should be the right basis set. In Figure 4 we see points that have undergone some transformation. We need to pick three points just like in the training stage to serve as the basis set.

After selecting a basis set we transform the other points using Equation (1) to obtain the index and subsequently the weighted and quantized index (a_s, b_s). We search for the presence of the index in the hash table and append a value of one if the index is found. We try this for every generated index by the remaining points in the scene image. The vote count is held by an accumulator. The accumulator is divided into various bins using a particular hash bin size. Each weighed and quantized index retrieves the information that is stored in its location in the hash table and a score is added to the retrieved information (model and basis set) found. The model and basis set with the highest values are considered as the estimate.

We experimented with algorithm and obtained the results shown in Figure 5. The results of the experiments are the recognition accuracy of the algorithm for different hash bin sizes. The plot shows when the right basis set is in either the top 10 or the top 20 of hypotheses generated by the voting.

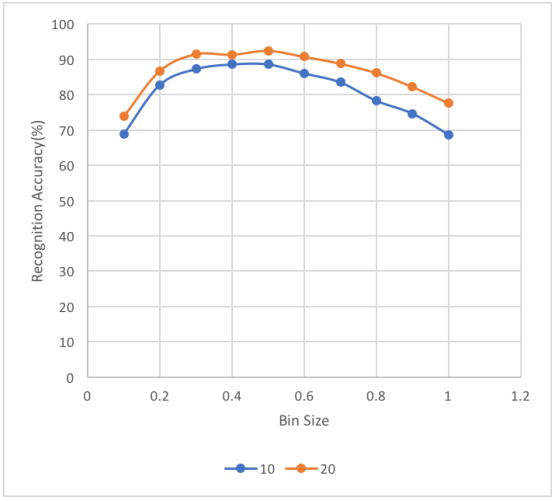

Fig. 5: The recognition accuracy.

We need a way to verify and refine the result. The refinement is an iterative process which uses a kd-tree to find the distances between the vertices and give each a neighbour. Then we solve the homography problem with RANSAC to find the optimal solution. We keep doing this until we find the error to within a certain tolerance level.

C. Viewpoint pose tracking

The movement of the camera relative to the pylon is tracked using an extended Kalman filter. For modelling the trajectory dynamics of the camera we use a random walk in the pose space.

1) State and Observation model: The random walk is based on the assumption that at each time the value of a variable is a random jump away from its previous value. The jumps are independently and identically distributed: $X_{n+1} = X_n + W_n$. Thus future values of variables depend on the previous observed values and noise.

We use the random walk model to track the movement of the camera. For this model the covariance grows linearly with time, which in many problems produces large dispersion. Also, there is no well-defined velocity for the object.

The pose is given as the components of the translation and rotation relative to the object to the camera. The pose has six degrees of freedom that fully quantify the position the camera. The translation can be represented as $t = [t_x, t_y, t_z]$ with a state transition of

$$t_{n+1} = t_n + v_n, \tag{4}$$

where v_n is a normal random variable with density $\mathcal{N}(0, \sigma^2 I)$.

The representation of rotation is difficult. The Euler angle and axis-angle representations tend to be easy but have a problem with singularities because of gimbal lock. To mitigate such issues we use a quaternion representation. The quaternion needs to meet the nonlinear constraint $q_0^2+q_x^2+q_y^2+q_z^2 = 1$. At each step there is normalization to ensure the constraint holds. We have a base unit quaternion q_0 representing the rotation, and a perturbation represented by q_p. At each incremental step we update the base quaternion to the estimated orientation and reset the axis-angle perturbation to zero. The base and perturbation quaternions combine to provide the full quaternion pose $q = q_0 q_p$. The transition model of the rotation is represented as

$$s_{n+1} = s_n + \epsilon_n, \tag{5}$$

where s represents the perturbation and $\epsilon_n \sim \mathcal{N}(0, \sigma^2 I)$. The combined state model for the full 6 degree-of-freedom pose is

$$x_n = \begin{pmatrix} t_n \\ s_n \end{pmatrix}. \tag{6}$$

The object is fixed while the camera is moving. Also, the 3D coordinates of the object are known and fixed. To obtain the camera dynamics we combine the calibrated intrinsic camera parameters K and the extrinsic parameters (R and t):

$$P = K[R|t].$$

The observation model is given as

$$z_n = h(x_n) + e_n. \tag{7}$$

The function $h(x_n)$ is nonlinear and combines the 3D coordinates and the camera projection.

$$h(x_n) = PX. \tag{8}$$

2) Data association: At each iteration of the tracker we have to ensure the vertices are matched properly. Therefore, we opt to use RANSAC to establish the correspondence. Using RANSAC also helps with handling of outliers.

3) Refinement: With the inliers obtained from RANSAC, sometimes the tracker falls into a local minimum which causes some drift. This drift eventually makes the tracker fail. We use an iterative technique for the refinement. The whole set of vertices is used for the refinement to ensure a better generalisation.

IV. Implementation and Results

We discuss the experiments performed and the results obtained for the algorithm. Geometric hashing serves as the initialization stage for the algorithm. Subsequently geometric hashing is only required if the tracking fails and new initial estimate of the pose is required.

As mentioned we created a model pylon for use in the experiments.

We initialized the pose estimate using geometric hashing, and then updated it via tracking using a Kalman filter. In both cases we used vertices from the neural network as the feature points.

For the geometric hashing we picked images from different views, and hand annotated them to serve as the models stored in the training stage. Each hand annotated vertex was provided with its corresponding 3D point as shown in Figure 6, also manually specified.

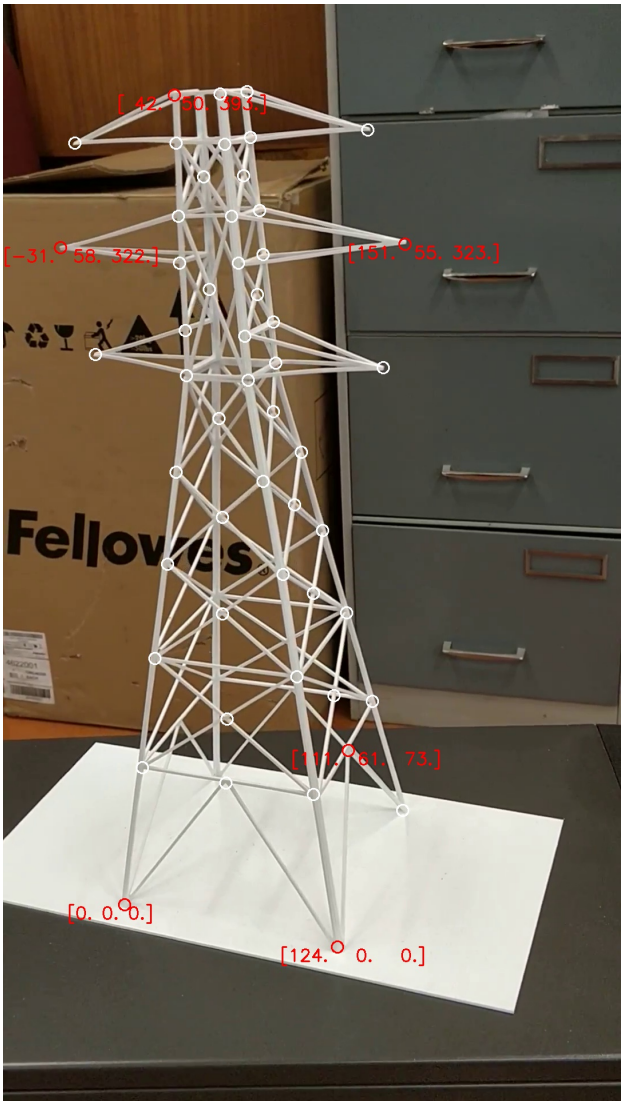

Fig. 6: Example of image used for training phase the geometric stage with some of the 3D coordinates.

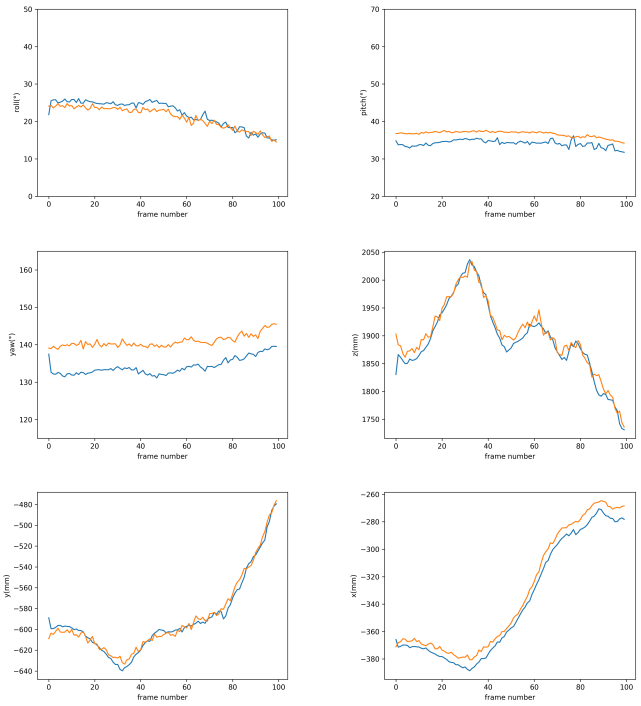

Fig. 7: The translation and orientation of the camera.

We used an affine transformation for the generation of the basis set and selected only non-collinear points.

Probalistic geometric hashing [17] takes 204s per frame, which is too slow. Therefore, we used the non-probabilistic voting technique because it is fast and relevant for a real-time system. The weight of the hash bin is given as 0.5 with a quantization of 0.25.

We tested the effectiveness of the algorithm with manual annotation of the four corners of the rectangular stand of the model pylon used to estimate the ground truth pose. Figure 7 and 8 provides information on the trajectory of the camera as it moves and takes images of the pylon. The Figure shows that the proposed methods succeed and are quite accurate, with mean camera position errors of 6.17mm, 0.89mm, 8.38mm for the x, y, and z directions, and mean errors of 1.04°, 2.53°, and 6.76° for the roll, pitch and yaw rotations.

Some snapshots of the results of the registration of the pylon is shown is Figure 9. We also provide video of the registration of the 3D model to the pylon image using the estimated pose of the camera obtained in each iteration at https://youtu.be/vqVtfkYOi8A.

The view of the pylon shows different vertices at any time. Some of these vertices are false and are due to the apparent crossings of the bars in the projected image. This makes the refinement and outlier rejection necessary. Even with fluctuating number of vertices the tracker is robust.

The effectiveness of the algorithm depends on how well the geometric hashing is able to detect the object either at the initialization or during relocalization of the detection. Therefore we attempted to see the rate of successful detection

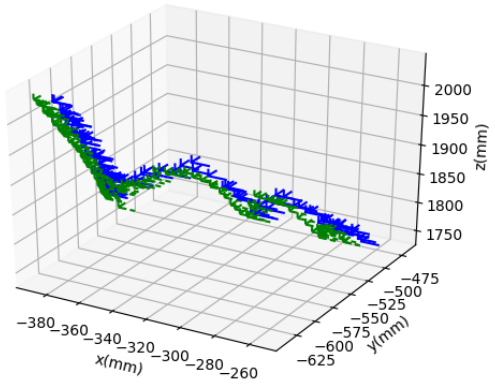

Fig. 8: The translation component of the camera pose in the x,y and z axis fused together when tracking. The blue represents the groundtruth and the green represents the estimate of the camera position.

for the geometric hashing over the full 100 frames, and obtained a result of 59%. The threshold of the root mean square error is set to 10 pixels. We select the first 50 basis set for the trial. During each trial, we generate 20 hypothesis candidates and pass them through the verification stage. In the verification stage, only candidates with a value less than the threshold and with a vote count of more than 25 in the accumulator are selected as accurate detections. An issue with geometric hashing is that it fails when the neural network falsely predicts a vertex and the vertex becomes part of the basis set.

We use a 11GB GTX 1080i GPU for the training of the neural network and takes 0.153s to produce the networkoutput. It takes 2.12s for pose estimation of the object using geometric hashing, and the time for each iteration of the Kalman filter is 0.204s.

V. Conclusion

We propose a method that can extract distinct features on a weakly textured wiry object such as a power pylon using a neural network, propose a method for pose estimation using geometric hashing, and formulate a Kalman filter tracker for subsequent incremental pose estimation. We show that it is possible to obtain a pose estimate of the camera and track the movement of the camera relative to the pylon. The algorithm can be used as part of an inspection system needed for the autonomous inspection of power line infrastructure.

References

[1] C. Sampedro, C. Martinez, A. Chauhan, and P. Campoy, "A supervised approach to electric tower detection and classification for power line inspection," in *2014 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2014, pp. 1970–1977.

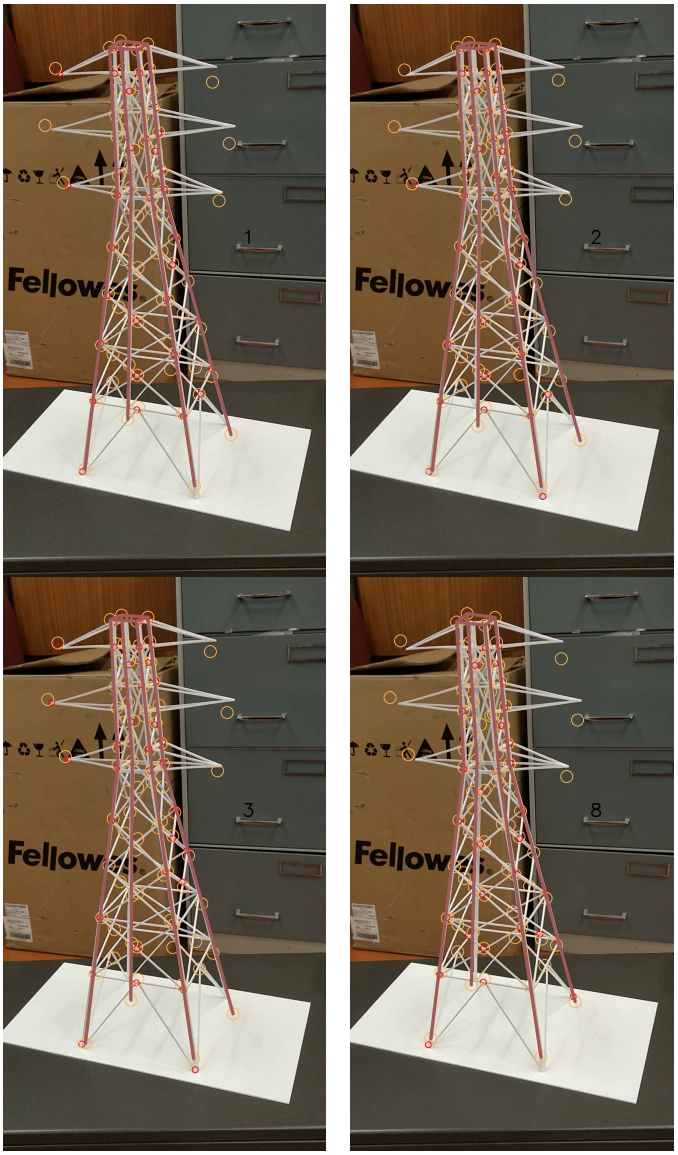

Fig. 9: Results of the 3D model of the pylon registering unto the image of the pylon.

[2] I. Golightly and D. Jones, "Corner detection and matching for visual tracking during power line inspection," *Image and Vision Computing*, vol. 21, no. 9, pp. 827–840, 2003.

[3] O. Araar, N. Aouf, and J. L. Vallejo Dietz, "Power pylon detection and monocular depth estimation from inspection uavs," *Industrial Robot: An International Journal*, vol. 42, no. 3, pp. 200–213, 2015.

[4] X. Wang and Y. Zhang, "Insulator identification from aerial images using support vector machine with background suppression," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2016, pp. 892–897.

[5] M. Oberweger, A. Wendel, and H. Bischof, "Visual recognition and fault detection for power line insulators," in *19th computer vision winter workshop*, 2014, pp. 1–8.

[6] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[7] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*. Springer, 2006, pp. 404–417.

[8] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski, "Orb: An efficient alternative to sift or surf." in *ICCV*, vol. 11, no. 1. Citeseer, 2011, p. 2.

[9] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *2011 international conference on computer vision*. IEEE, 2011, pp. 858–865.

[10] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3d object recognition," in *2010 IEEE computer society conference on computer vision and pattern recognition*. Ieee, 2010, pp. 998–1005.

[11] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6d object pose estimation using 3d object coordinates," in *European conference on computer vision*. Springer, 2014, pp. 536–551.

[12] A. Crivellaro, M. Rad, Y. Verdie, K. M. Yi, P. Fua, and V. Lepetit, "Robust 3d object tracking from monocular images using stable parts," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 6, pp. 1465–1479, 2017.

[13] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele, "Deepercut: A deeper, stronger, and faster multi-person pose estimation model," in *European Conference on Computer Vision*. Springer, 2016, pp. 34–50.

[14] A. Mathis, P. Mamidanna, K. M. Cury, T. Abe, V. N. Murthy, M. W. Mathis, and M. Bethge, "Deeplabcut: markerless pose estimation of user-defined body parts with deep learning," Nature Publishing Group, Tech. Rep., 2018.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[16] H. J. Wolfson and I. Rigoutsos, "Geometric hashing: An overview," *IEEE computational science and engineering*, vol. 4, no. 4, pp. 10–21, 1997.

[17] I. Rigoutsos and R. Hummel, "A bayesian approach to model matching with geometric hashing," *Computer vision and image understanding*, vol. 62, no. 1, pp. 11–26, 1995.