Simple Stereo Matching Algorithm for Localising Keypoints in a Restricted Search Space

Matthew Seabright, Lee Streeter, Michael Cree, Mike Duke, Rachael Tighe
School of Engineering
University of Waikato
Hamilton, New Zealand
mjseabright@gmail.com

***Abstract*—Modern stereo matching algorithms generally rely on matching image features such as colour and texture to find corresponding matches between images. They can provide very good results but be very computationally intensive. However, in cases where the objects to be localised lie approximately on a known plane, a much simpler algorithm can be applied. One such case is localising kiwifruit in modern orchards, as the plants are trained to grow in a planar structure known as a pergola. The proposed algorithm uses tight distance limits (based on orchard geometry) to reduce the search space for matching fruit to a small window. For the majority of kiwifruit, the search window is small enough to contain only one fruit in the adjacent image, giving only a single solution. In cases where there are tightly grouped fruit or false positive fruit detections adjacent to true positive fruit detections, there can be multiple potential matching solutions. To solve these, each potential solution is evaluated based on how closely it conforms to the mean object distance from camera and a solution is selected. On real world test data containing 121 image pairs, the algorithm has a 99.2% true positive rate. Computation time was 1.97 ms per image pair.**

***Index Terms*—Stereo matching, kiwifruit, detection, convolutional neural networks,**

I. Introduction

Kiwifruit is New Zealand's largest horticultural export bringing in NZ$2.23 billion in 2016/2017 [1] and expected to rise to NZ$4.5 billion by 2025 [2]. The industry is already struggling to find the required labour to meet this demand [3]. Automated systems are needed to reduce human labour requirements and increase productivity. Yield estimation is one of the tasks that has been identified as a candidate for automation. Accurately counting fruit for yield estimation requires fruit to be identified and localised within an orchard.

A common technique used for localising fruit is stereo vision. A detection system is used to detect the fruit in each image of the pair, then a matching algorithm is applied to perform matching, allowing localisation. Current convolutional neural networks provide a high performance detection system, despite their high computational intensity. Stereo matching, or the correspondence problem, is identifying which fruit in the left image and right image represent the same fruit in the real world.

The more that is known about the scene, the more the complexity of the matching problem can be reduced. The data used for this paper is taken from kiwifruit orchards where the aim is to localise fruit for the purposes of counting. Kiwifruit are grown using a pergola growing system which has the kiwifruit situated in a canopy between 1.6 and 2.1 metres from the ground. When localising kiwifruit calyxs[1] that have been imaged from below, the range of distances the fruit are likely to be situated from the camera plane is therefore known. The algorithm presented makes use of this knowledge to reduce the search space for a match to a small area of the image. For the majority of cases, the reduced search space results in no ambiguity in matching, that is, there is only one possible matching combination that satisfies the matching criteria. In the cases there is ambiguity, all potential matching combinations are evaluated based on their conformance to the mean distance from the camera plane of all previously seen matches. In this work the keypoint matching algorithm is presented and its performance evaluated on a dataset of kiwifruit images.

II. Related Work

For use in an apple harvester, an area based matching algorithm was used [4]. Detection was performed on each image and binary thresholding applied to mark all apple pixels. Fruit were matched based on epipolar geometry and finding detected fruit with similar area in each image. An ordering constraint was applied to ensure that matched fruit appeared in the same order along the x-axis in both images. The method achieved a 95% success rate with the authors noting failures mostly caused by apples overlapping in one image but not the other, causing differences in measured area.

To localise tomatoes and apples, a mean disparity approach was used [5], [6]. First, a matrix was constructed that contained the disparity (in the x-axis) of every possible matching combination. All values that fell outside disparity thresholds (decided by the distance they expected fruit to be from the cameras) were then removed. The mean disparity was calculated and

This research was supported by the New Zealand Ministry for Business, Innovation and Employment (MBIE) on contract UOAX1414.

978-1-7281-0125-5/18/$31.00 ©2018 IEEE

[1]The calyx of a kiwifruit is the dark, fury bit opposite where the stem attaches.

for fruit with multiple potential matches, the match with the disparity furthest from mean was removed. This was repeated until only one or zero matches were left for each detected fruit. Neither works quantified the performance of their systems.

For localising peach blossoms, a trinocular camera setup was used [7]. Their matching algorithm can utilise all three cameras or just two, based on which combination gives the best quality match. To evaluate matches, a fixed size window around a detected blossom was extracted and compared the pixels to detected blossoms in the other images. They perform the comparison at multiple window locations to find the location of highest correspondence between the images. Their method is very computationally intensive, taking 60-80 seconds per image pair, however they are using very high resolution images (10 MP) and have very high number of blossoms per image.

For use in localising litchi, a gray-scale correspondence matching algorithm was employed [8]. All of the pixels belonging to a single litchi in one image would be swept along the epipolar line in the other image. The similarity of the grey value of the pixels in the two images was evaluated at each point along the epipolar line using a normalised cross-correlation. A match was then declared at the point of maximum similarity. Their algorithm was able to correctly match 98% of unoccluded litchi and 94% of partially occluded litchis. The authors note that the main reason for mismatching partially occluded litchis was the shape of the litchi appeared different in the two images. The shape difference caused caused a mismatch in some cases.

For use in a kiwifruit harvester, a reduced search space template based algorithm was used [9]. A fixed size template was was taken from each detected kiwifruit calyx and converted to gray-scale. For each calyx, a search window was identified in the other image of the stereo pair based on camera geometry and expected object distance. The template was then swept through the search window and the sum of squared differences between the overlapping patch of image and the template calculated for each position. The position with the lowest sum was selected as the matching point, if it met a set threshold. The author cites computation time as 14-26 seconds per image pair. However this was run on a CPU from 2007 and programmed without regard for computation time.

The methods outlined above all produce acceptable matching accuracy when used on their respective datasets. However, those that reported computation times showed that their algorithms are very computationally intensive. This high computation time is due to many comparisons being run on subsets of the images. With the relatively 2D nature of kiwifruit orchards, a simpler algorithm could be applied that forgoes comparing features of the image. Such an algorithm could offer both high accuracy and very low computation times.

III. Algorithm Description

A keypoint is a point on an image that corresponds to the center of an object in a scene. A detection system, such as a convolutional neural network, is used to detect the object of interest in each image of a stereo pair. A stereo keypoint matching algorithm is then applied to correspond a keypoint in one image to a keypoint in the other. The location of the object in 3d space can then be infered using the camera geometry.

The stereo keypoint matching algorithm has two prerequisites. The first prerequisite is the calibration matrices for both the individual cameras as well as the camera pair. These matrices are obtained using the standard checkerboard calibration routines in OpenCV [10]. The second prerequisite is size and position of the search windows, see Fig. 1. For each keypoint, there is a corresponding search window that defines the area of the other image where the match should be found. The search window is centred on the epipolar line. The camera geometry and object distances define the position and dimensions of the search windows. The width and position along the epipolar line are defined by both the camera geometry and the object distances. Ideally, the search window would have a height of one pixel as a feature in one image should lie exactly on the corresponding epipolar line in the other image. However, detection systems are not perfect, especially when an object is partially occluded in one or both images. There can also be inaccuracies in the camera calibration, therefore, the height of the search window should be large enough that small errors in the location of keypoints do not prevent matches. Both the width and height of the search window can be increased to help account for these inaccuracies, but the larger the search window the higher the chances of an incorrect match and the more computationally intensive. The dimensions and position of the search windows are calculated via the use of epipolar geometry with OpenCV.

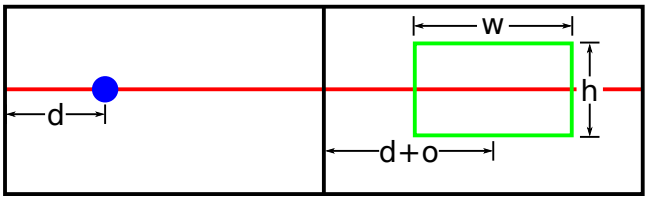

Fig. 1. The geometry of a keypoint and corresponding search window in the left and right images. The red line is an epipolar line running through the rectified and undistorted images. The blue dot is a keypoint in the left image. The green box is the search window in the right image where a keypoint matching the blue dot should be found. The width of the search window, `w`, is defined by the camera geometry and the expected range of object distances. The position of the search window along the epipolar line is the position of the keypoint along the epipolar line, `d`, plus an offset, `o`. The offset, `o`, is defined by the camera geometry and the range of object distances. The height of the search window, `h`, is made large enough to account for errors in camera calibration and the detection system.

Once the above prerequisites are met, the algorithm is applied. The algorithm will only match two keypoints if both the keypoints are contained by the other's search window. The algorithm consists of two sections. The first is the 'easy' part which handles unambiguous matches. Unambiguous matches are matches where both the keypoints are exclusively contained by the others search window. For example, in Fig. 3c-d, keypoint 8 is the only keypoint contained by search window 37 and keypoint 37 is the only keypoint contained by

search window 8, therefore, the only matching combination that meets the requirements is keypoint 8 with keypoint 37. The second section of the matching algorithm is the 'difficult' part which handles ambiguous matches. Ambiguous matches are matches where a keypoint is contained by multiple search windows, or there are multiple keypoints contained by a single search window. For example, in Fig. 3c-d, keypoints 6 and 7 are both contained by search windows 0 and 17 and keypoints 17 and 0 are both contained by search windows 6 and 7. Keypoint 7 could be matched with either keypoint 0 or 17 while maintaining the matching requirements that have been set, hence, there is ambiguity about which keypoint should be matched with which. The 'difficult' section decides which matching combination is most likely correct by evaluating each potential combination. The evaluation judges matches based on their conformance to the mean distance from the camera of all previously seen matches (both in the current image, and in past images). To do this, a mean of the position of each keypoint within its matching search window is kept, see Fig. 2. Two of these means are maintained, one for all previously seen left images, and the other for all previously seen right images. These means are referred to as the 'mean position in window' henceforth.

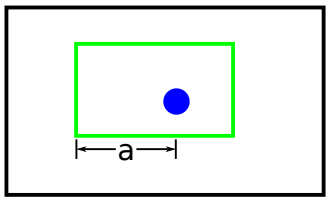

Fig. 2. A keypoint (blue) in its matching search window (green). The distance, a, from the left edge of the search window to the keypoint is logged for both the right image and left image. The mean of all previously matched keypoints is calculated and used to evaluate ambiguous matches in the 'difficult' matcher.

The 'easy' matcher finds all the matches where there is only a single keypoint in a search window, and the corresponding keypoint is also in the corresponding search window in the other image. These matches are unambiguous as there is only one possible matching combination that meets the restrictions set by the camera geometry and the distance thresholds set. For each keypoint in an image, a search window is created in the other image (Fig. 3c-d). A list of all of the search windows and keypoints is formed. All the search windows in the left image are checked to find all the keypoints that lie within each search window. If there are no keypoints in a search window, there is no match, so the window and the corresponding keypoint (from the right image) are removed from the list. If there is only one keypoint in a search window, all other search windows are checked to see if they also contain that keypoint. If none do, the corresponding keypoint and search window in the right image are checked to see if the keypoint is contained in the search window. If they do, it is a match and the keypoints and search windows are removed from the list. For both the left and right image, the distance between the keypoint and the left edge of the search window containing that keypoint is recorded and the mean position in window is updated, see Fig. 2. All of the unambiguous matches have been found. If there are any remaining keypoints in either image that do not lie within a remaining search window, they are removed from the list.

The 'difficult' section of the matcher now matches all the remaining keypoints. Each keypoint and search window is assigned to a group of ambiguity, see Fig. 4. If a search window contains multiple keypoints, the keypoints, the search window and all of their corresponding keypoints and search windows from the other image are put into one group of ambiguity. This is continued recursively until all remaining keypoints are assigned to a group of ambiguity. For example, in Fig 3e-f, keypoints 7, 17, 6 and 0 would form the only group of ambiguity. The algorithm counts the number of keypoints per image in each group. If there is a mismatch in the number of keypoints in each image, 'dummy' keypoints are added to the group to correct the mismatch. The 'dummy' keypoints have a corresponding search window that contains the entire image, meaning they will contain all of the keypoints, and hence, can be matched with any of the keypoints. If a real keypoint is matched to a 'dummy' keypoint, it represents no match being found for the real keypoint.

Every possible matching combination of the keypoints is listed. A possible match is when a keypoint lies within a search window, and the corresponding keypoint in the other image lies within the corresponding search window. For example, the two possible matching combinations in Fig. 3e-f are;

- 7 matched with 17 and 6 matched with 0
- 7 matched with 0 and 6 matched with 17

For each match in a potential matching combination, the distance between the keypoint and the left edge of the search window that the keypoint is being matched with is calculated and subtracted from the mean position in window. The absolute value of each of these differences is summed for all of the matches in a potential matching combination. For the example above (Fig. 3e-f) the two sums would be:

$$s_1 = |M_l - a_{7,17}| + |M_r - a_{17,7}| + |M_l - a_{6,0}| + |M_r - a_{0,6}|$$

$$s_2 = |M_l - a_{7,0}| + |M_r - a_{0,7}| + |M_l - a_{6,17}| + |M_r - a_{17,6}|$$

where M_l is the left image mean position in window, and $a_{7,17}$ denotes the distance between keypoint 7 and the left edge of search window 17. The potential matching combination with the lowest sum is selected and the matches logged. The distance between the keypoint and the left edge of the search window containing the keypoint is logged and the mean position in window updated.

IV. Evaluation

Image data was collected from a quad bike with four Ace acA1920-40uc cameras (Basler, Ahrensburg, Germany) with LM12HC lenses (Kowa, Aichi, Japan) arranged as two stereo pairs. The cameras and an LED lighting system face upwards to the kiwifruit canopy. The camera pairs have a baseline of 120 mm and a spacing of 750 mm between the two pairs.

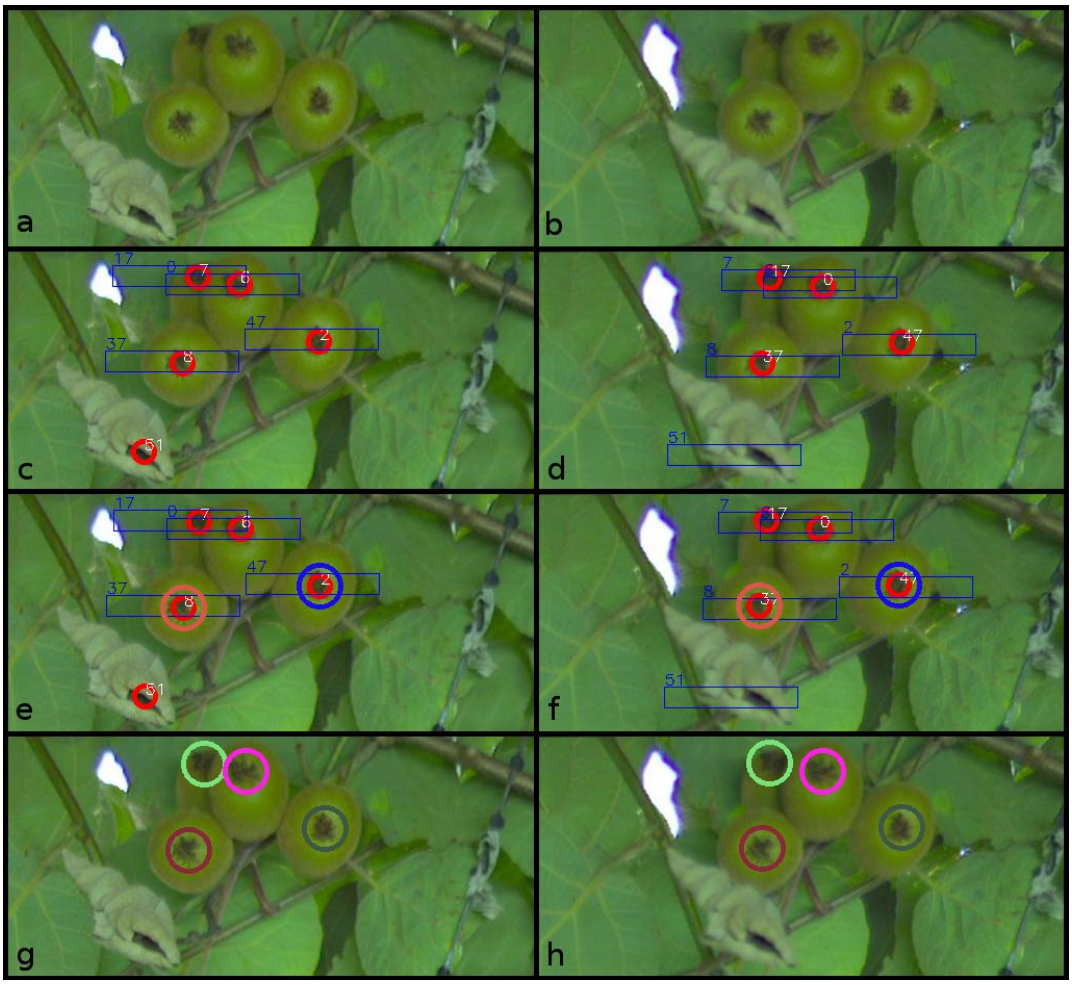

Fig. 3. The steps of the matching algorithm. Panels a and b show a subsection of images from the left and right cameras of a stereo pair. Panels c and d show the same images with the keypoints marked as red circles, (detected as calyxes by the detection system) and the corresponding search windows marked as blue rectangles. Notice the number on the calyx labels correspond to the numbers on the search windows in the opposite image. There is also a false positive detection shown in the left image. Panels e and f show that two of the fruit have been correctly matched. These were matched by the 'easy' part of the algorithm as there is no ambiguity caused by search windows containing multiple keypoints. The remaining four keypoints (two in each image) form a group because of their overlapping. Each potential matching combination is evaluated and the combination that most closely conforms to the mean height of all fruit previously matched is chosen. Panels g and h show the final result with the correct matches being found for all four fruit.

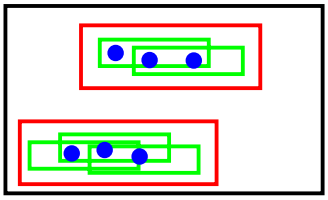

Fig. 4. Two groups of ambiguity. The blue dots are keypoints in an image. The green rectangles are search windows. The red rectangles show how the 'difficult' matcher would group the keypoints and search windows. Within each group, there is ambiguity as to which search window should be matched with each keypoint.

The rig was taken through 25 green kiwifruit orchard blocks around Tauranga, New Zealand, that use the pergola growing system. While travelling at 5 km/h, each of the four cameras were simultaneously triggered at 7 frames per second with exposure time of 3 ms. Images are saved via a laptop mounted on the quad bike. Three passes were taken down each row of the orchard, one down each side and one down the middle. Multiple passes gives full coverage of the row despite the large range of row widths encountered in kiwifruit orchards. Over the course of two weeks in January 2018, approximately 1.4 million image pairs were captured.

The cameras were calibrated using the OpenCV checkerboard pattern. Images were taken of the checkerboard in a range of positions and orientations throughout the field of view of the cameras. The OpenCV functions `calibrateCamera` and `stereoCalibrate` were then used to produce the relevant matrices describing each camera and the geometry of the two camera pairs.

Of the captured images, 109 were randomly selected and labelled with bounding boxes for each kiwifruit calyx. Tensorflow was used to train an implementation of Mask RCNN

[11]–[14] using a GTX Titan Xp (Nvidia, Santa Clara, USA). Pre-trained weights were used and training was performed on 69 of the labelled images with 20 being used as a validation set. Inference was performed on all of the 2.8 million images. The centre of each of the detected calyx bounding boxes was used as the keypoint for the stereo matching. Inference performance was measured across 20 images (not used for training) with a mean average precision (mean of the average precision for each of the 20 images) of 0.898 and mean recall of 0.909 at an intersection over union of 0.5.

The stereo matching algorithm was implemented in the Python programming language. The valid range of fruit distances from the cameras was set to 900-1700 mm. When combined with the camera geometry used, this range of distances corresponds to an offset of 207 pixels and a search window width of 128 pixels (o and w in Fig 1. respectivly). Seach window height (h in Fig 1.) was set to 20 pixels. The algorithm was run on 121 randomly selected image and the results manually evaluated. Matches were classified as 'correct' if a fruit in the left image was matched to the same fruit in the right image. Matches were classified as 'incorrect' if a fruit in the left image was matched to a different fruit in the right image. Matches were classified as 'false positive' if two false positive detections were matched to each other. The matching algorithm was run over 5000 random images and the execution of the matching algorithm was timed. The mean time for each image pair was calculated. Trails were run on a PC with an Xeon E5-1650 V3 (Intel, Santa Clara, USA) CPU and 16 GB of ram.

V. Results & Discussion

A total of 3768 matches were found (31.1 matches per image pair) across the 121 image pairs evaluated, Tab. I. The correct match was found in 99.23% of cases and incorrect matches in 0.16%. False positive detections accounted for the other 0.61% of cases and are solely due to the detection system misclassifying areas of the image (Fig. 5).

TABLE I
Matching results

	Occurrences	**% of total matches**
Image pairs	121	-
Total matches	3768	100 %
Correct matches	3739	99.23 %
Incorrect matches	6	0.16 %
False positive matches	23	0.61 %

In five of the six cases of incorrect matches, there was a tight cluster of fruit where all fruit were detected in one image, but one fruit was missed in the other image (Fig. 6). When fruit in a cluster that is at a height significantly different from the mean fruit height are missed, an incorrect matching combination can be selected. The number of missed fruit could be reduced by improving the detection system, preventing incorrect matches occurring. The number of incorrect matches could also be reduced by adding a step to compare how similar the fruit look across the two images. However, adding steps would increase computation time and complexity. Another potential solution is to not just use the global mean fruit height as the deciding metric but also use the height of the fruit nearby. A nearby fruit height mean could help as fruit height tends to change gradually across kiwifruit orchards rather than very suddenly.

The other case of an incorrectly matched fruit was caused by two adjacent, low hanging fruit that were outside the imposed height range bounds. The fruit being out of bounds meant that the correct matching combination was deemed invalid as the keypoints fell outside the correct search windows. Because of the arrangement of the fruit, an incorrect matching combination did meet the matching criteria of keypoints lying in search windows and was therefore selected (Fig. 7). To avoid out of bounds fruit causing incorrect matches, the size of the detected object in the image could be used as a filter to remove very close or far away objects. However size filtering would only be effective in situations where there is relatively small variation in object size and could lead to a decrease in accuracy in some cases. Overall, however, any further changes should be carefully weighed against the goal of maintaining the already high matching success rate

Over 5000 image pairs, a mean of 1.97 ms per pair was needed to perform matching. Computation time could be improved by evaluating matches in parallel, taking advantage of modern multi-core CPUs. Further performance gains could potentially realised by using a more efficient language such as C++ rather than Python that was used for this evaluation.

The algorithm described is best suited to quickly localising objects in environments that are largely structured but contain variation. Horticultural applications such as harvesting, spraying, weeding, counting and monitoring are a good fit. Other applications such as aerial surveying, certain manufacturing tasks and automated sports analysis could also be suitable.

VI. Conclusion

A stereo keypoint matching algorithm has been presented and shown to be effective at matching detected keypoints in real world test data with a success rate of 99.23% and mean computation time of 1.97 ms. The incorrect matches encountered were largely due to clustered fruit not being detected. Improvements have been suggested to lower computation time and prevent some of the incorrect matches. The algorithm presented is a positive step towards solving the labor shortage problems seen in the New Zealand Kiwifruit industry.

Acknowledgment

The authors would like to thank Plus Group Horticulture and the orchard owners for providing access to their orchards, Robotics Plus for providing resources and assiatance, and all of the Cohort project members from The University of Auckland, University of Waikato and Plant and Food Research.

References

[1] Plant & Food Research, "Fresh Facts," 2016. [Online]. Available: http://www.freshfacts.co.nz/files/freshfacts-2016.pdf [Accessed 2018-09-13]

Fig. 5. Example of a false positive match. The match marked in green is not a fruit, but a dead leaf. The detection system has misclassified the leaf as a calyx in both images and the matching algorithm has matched it.

Fig. 6. Example of an incorrectly matched fruit. The match marked in gray is not the same fruit in both images. There are three fruit in a cluster with the middle fruit being partially occluded by the other two. All three of the fruit were detected in the right image, but the middle fruit was missed in the left image. The missed detection resulted in the wrong match being found.

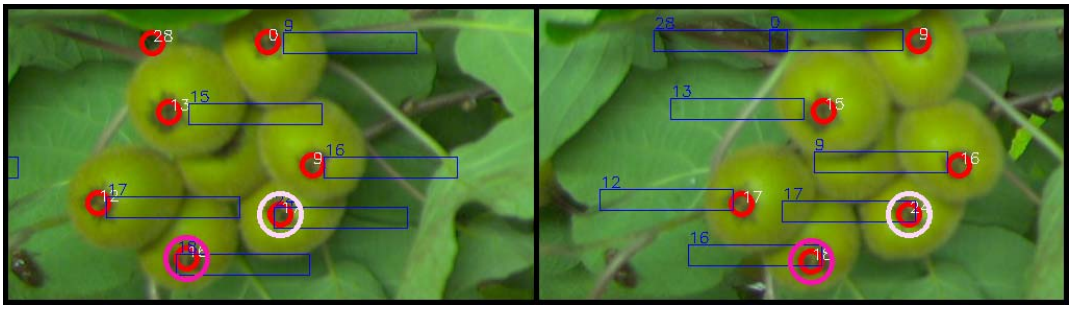

Fig. 7. Example of fruit not being matched because they are too close to the camera plane. The keypoints fall outside the search window that they would be in if they were within the valid height range. Detected objects that lie outside the distance bounds will either not be matched or will be matched incorrectly.

[2] Zespri International Ltd., "ZESPRI Annual Report 2016-17," Tauranga, New Zealand, 2017. [Online]. Available: http://www.zespri.com/ZespriInvestorPublications/Annual-Report-2016-17.pdf [Accessed: 2018-09-13]

[3] D. Picken, "Inside the Bay of Plenty's kiwifruit labour shortage," Bay of Plenty Times, 2018. [Online]. Available: https://www.nzherald.co.nz/nz/news/article.cfm?c_id=1&objectid=12053443 [Accessed: 2018-09-13]

[4] Y. Si, G. Liu, and J. Feng, "Location of apples in trees using stereoscopic vision," *Computers and Electronics in Agriculture*, vol. 112, pp. 68–74, 2015.

[5] A. Plebe and G. Grasso, "Localization of spherical fruits for robotic harvesting," *Machine Vision and Applications*, vol. 13, no. 2, pp. 70–79, nov 2001.

[6] R. Xiang, H. Jiang, and Y. Ying, "Recognition of clustered tomatoes based on binocular stereo vision," *Computers and Electronics in Agriculture*, vol. 106, pp. 75–90, 2014.

[7] M. Nielsen, D. C. Slaughter, and C. Gliever, "Vision-based 3D peach tree reconstruction for automated blossom thinning," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 1, pp. 188–196, 2012.

[8] C. Wang, X. Zou, Y. Tang, L. Luo, and W. Feng, "Localisation of litchi in an unstructured environment using binocular stereo vision," *Biosystems Engineering*, vol. 145, no. May 2016, pp. 39–51, may 2016.

[9] A. J. Scarfe, "Development of an Autonomous Kiwifruit Harvester," Doctral Thesis, Massey University, 2012. [Online]. Available: https://mro.massey.ac.nz/bitstream/handle/10179/4426/01_front.pdf

[10] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[11] W. Abdulla, "Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow," 2017. [Online]. Available: https://github.com/matterport/Mask_RCNN

[12] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *2017 IEEE International Conference on Computer Vision (ICCV)*. Venice, Italy: IEEE, oct 2017, pp. 2980–2988.

[13] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[14] R. Girshick, "Fast R-CNN," in *2015 IEEE International Conference on Computer Vision (ICCV)*, vol. 2015 Inter. Santiago, Chile: IEEE, dec 2015, pp. 1440–1448.