Computers and Electronics in Agriculture 206 (2023) 107688

Contents lists available at ScienceDirect

Computers and Electronics in Agriculture

journal homepage: www.elsevier.com/locate/compag

Original papers

Multi-view triangulation without correspondences

Mathieu Gaillard [a], Bedrich Benes [a,*], Michael C. Tross [b,c], James C. Schnable [b,c]

[a] Department of Computer Science, Purdue University, West Lafayette, 47906, IN, USA
[b] Center for Plant Science Innovation and Department of Agronomy and Horticulture, University of Nebraska–Lincoln, Lincoln, 68588, NE, USA
[c] Complex Biosystems Graduate Program, University of Nebraska–Lincoln, Lincoln, 68588, NE, USA

ARTICLE INFO

Keywords:
High-throughput phenotyping
Multi-view 3D reconstruction
Sorghum leaf counting

ABSTRACT

We introduce a novel method to solve the joint problem of correspondence and triangulation of points from multiple calibrated perspective views and show its application to counting the number of leaves present on plants photographed from multiple angles in phenotyping facilities. Assuming there is a set of n calibrated views of an object, the input to our algorithm is a set of 2D points (*e.g.,* leaf tips, fruits, flowers) detected in each of the n views, and the output is a set of 3D points corresponding to the 2D points when re-projected on views. Our approach is robust to noise and occlusion. In particular, it is not required for all points to be visible from one of the views, as our algorithm can infer that a point is occluded by reasoning on the 3D geometry of the scene. Our algorithm is suitable for many points (approx. thousands) reconstructed from a reduced set of views (up to 15). For example, our implementation finds the correspondence of 20 points captured by six cameras in about one second on consumer hardware. We evaluate the performance on synthetic data as well as real examples. We show that the accuracy of leaf counting from multi-view images is drastically improved by our algorithm.

1. Introduction

Counting and tracking organs and other regions of interest (*e.g.,* leaves, flowers, fruits, branches, disease lesions) of a plant over time is a fundamental problem faced by the phenotyping efforts that support both plant breeding and production agriculture. Plant development is quite plastic relative to animal development. The number of leaves observed at maturity in genetically distinct maize plants grown in the same environment varied from 8 to 21. Even two genetically identical plants can produce varying number of organs, particularly when grown in different environments, as shown for wheat Brooking et al. (1995) or maize (Tollenaar and Hunter, 1983). Knowing the timing of emergence for each leaf and how many leaves a plant will have produced at maturity are critical parameters for a number of crop growth models (Lizaso et al., 2003; Hammer et al., 2010; He et al., 2012; Truong et al., 2017).

However, counting and tracking plant organs is a time-consuming and error-prone process. To address this issue, researchers have built high-throughput phenotyping facilities (Fahlgren et al., 2015; Junker et al., 2015; Ge et al., 2016) that automatically take photos of hundreds of plants from a small number of fixed angles (*e.g.,* 5–10 sides and one top view). Recent work (*e.g.,* Gaillard et al., 2020) has shown that the image data generated by these facilities are suitable for plant 3D reconstruction. Although phenotyping facilities are convenient because they automate data acquisition, state-of-the-art automatic leaf counting is still not as accurate as human labor. For instance, recent research from Miao et al. (2021) presents a deep neural network that shows near-human accuracy in counting maize and sorghum leaves by automatically detecting leaf tips. But this approach – both when automated via a deep neural network and when humans count leaves from an image – fails to account for the inherent 3D structure of plants. As a result, only leaves visible from a single perspective are counted, even when multiple images from different perspectives are collected. The single-view approach fails when not all organs or other regions of interest are visible due to occlusion. Unfortunately, occlusion is quite common and can both reduce accuracy and introduce bias into both single time point data and when phenotypes are scored and tracked over time.

We focus on the general problem of finding corresponding points in multiple images of a 3D object (the correspondence problem) and estimating their 3D positions based on their projections on calibrated views (the triangulation problem). Solving the correspondence and triangulation problems from multiple views significantly improves the capability and utility of plant phenotyping for a range of plant traits. It enables counting the number of organs or other regions of interest from multiple views and accurately estimating their 3D positions. This

[*] Corresponding author.
E-mail address: bbenes@purdue.edu (B. Benes).

https://doi.org/10.1016/j.compag.2023.107688
Received 16 August 2022; Received in revised form 6 December 2022; Accepted 28 January 2023

can be used to track and estimate traits related to canopy architecture, like phyllotaxis, that can, in turn, create variation in traits like photosynthetic efficiency and plant water usage.

We introduce the following workflow: (1) we assume that we have as input a set of at least two calibrated plant views, and (2) we know the 2D locations of some organs or other regions of interest (*e.g.,* leaf tips, branching points, fruits, flowers, disease lesion) in each view. The annotation of the locations of organs or other regions of interest can result from either manual annotation or automatic detection via deep learning models. The output is a set of 3D points corresponding to the 2D points when re-projected on views.

We propose a novel solution to the joint problem of correspondence and triangulation of a set of 3D points projected on multiple calibrated views. Our method works in the presence of noise and occlusion and with a low number of views. Pairs of views can be highly different if the cameras are arbitrarily located *i.e.,* behind the object. We do not assume a rigid motion of the plant between views so our method is robust to parts of the plant slightly moving between views. Our algorithm does not require feature detection (*e.g.,* with SIFT Lowe, 2004) to describe the points in images; instead, it only relies on the scene geometry *i.e.,* the camera estimated poses. The only hyper-parameter of our algorithm is a threshold θ in pixels beyond which two 2D points cannot be matched after re-projection. We use dynamic programming to solve the NP-complete problem of sparse multi-view correspondence. Dynamic programming decreases the time complexity at the expense of memory consumption, and our implementation makes our solution suitable for solving problems with thousands of points seen from up to 15 views.

Our validation shows that our method drastically improves the current state-of-the-art sorghum leaf counting accuracy from multi-view images captured in a phenotyping facility. The key advantage of our approach is that it can aggregate the information from multiple views and detect all phenotypes in 3D space, even if they are not all visible from one of the 2D views. This robustness to occlusion and false negatives makes our algorithm particularly powerful when paired with a detection neural network like described by Pound et al. (2017) and Miao et al. (2021).

2. Algorithm development

2.1. Problem formulation

Fig. 1 shows the schematics of the used notation. Three views V_1, V_2, and V_3 observe two points Q_1 and Q_2. Because of the presence of the occluder, V_2 cannot see Q_2 and V_3 misses Q_1. The rays r_{11} and r_{12}, cast from the first view, are not occluded. Therefore V_1 can see both points.

Let us have a collection of k calibrated views $\mathcal{V} = \{V_1, V_2, \dots, V_k\}$ with projection matrices denoted by $\{P_1, P_2, \dots, P_k\}$. Each view includes a set of n_k 2D points $\mathcal{P}_k = \{p_{k,1}, p_{k,2}, \dots, p_{k,n_k}\}$. Note that because of potential occlusion, the number n_k of points per view can vary. We seek to find a set of n 3D points $\mathcal{Q} = \{Q_1, Q_2, \dots, Q_n\}$ that correspond to the original points projected on $\mathcal{V}$.

We introduce $\Psi = \{\psi_1, \psi_2, \dots, \psi_n\}$ a set of n bijective maps $\psi_n = \{p_{i,j} \mid p_{i,j} \text{ corresponds } to\ Q_n\}$ between a 3D point q and at most one 2D point p from each of the k views. In other words, for all views in $\mathcal{V}$, each 2D point $p_{i,j}$ from view V_i is associated to one 3D point from $\mathcal{Q}$. Although Ψ is not the solution to the problem, it is an essential transitional building block that retains the correspondences between points in projected views and the 3D points. It can be used to triangulate 3D points $\mathcal{Q}$:

$$\mathcal{Q} = \{\Delta(\psi_i) \mid \forall i \in [1, n]\} \tag{1}$$

where the $\Delta(\psi)$ is a triangulation function that takes as input a set of correspondences ψ and outputs the triangulated 3D point in $\mathcal{Q}$. Note that the triangulation function Δ is specific to the scene geometry and depends on the calibration of views in $\mathcal{V}$.

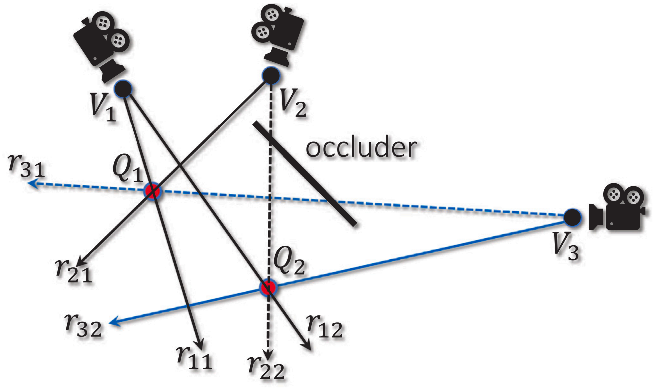


Fig. 1. Schematics of three views V_1, V_2, and V_3 observing two points Q_1 and Q_2. The dashed rays r_{31} and r_{22} indicate that the point cannot be visible because of the occlusion. In this example, V_1 and V_2 are already matched together, and V_3 is being matched to the result. The result from $\mathcal{S}(\{V_1, V_2\})$ is $\mathcal{Q} = \{Q_1\}$ and $\Psi = \{\{p_{1,1}, p_{2,1}\}, \{p_{1,2}\}\}$. During the execution of $\mathcal{M}(V_3, \{\mathcal{Q}, \Psi\})$, the 2D point $p_{3,2}$ is matched to the 2D point $p_{1,2}$ because $sf_l(p_{3,2}, p_{1,2}) < sf_p(p_{3,2}, Q_1)$.

Ultimately, we want to find the set of 3D points $\mathcal{Q}^*$ that best represents the input 2D points $\mathcal{P}$. This set $\mathcal{Q}^*$, and its associated correspondence maps Ψ^*, minimize the re-projection error E of points in $\mathcal{Q}^*$ on the calibrated views $\mathcal{V}$:

$$E(\Psi) = \sum_{m=1}^{n} \sum_{p_{i,j} \in \psi_m} \left\| p_{i,j} - P_i \cdot \Delta(\psi_m) \right\|_2 \tag{2}$$

where $p_{i,j}$ is the 2D position of the jth point on view V_i and $P_i \cdot \Delta(\psi_m)$ is the 2D projection of the 3D point $Q_m = \Delta(\psi_m)$ on view V_i:

$$\begin{aligned} \Psi^* = \ & \underset{\Psi}{\arg\min} \quad E(\Psi) \\ & \textit{subject to } \Psi \notin \emptyset \\ & \qquad \forall p_{i,j} \in \mathcal{P}, \exists m \text{ s.t. } p_{i,j} \in \psi_m \\ & \qquad \forall m, \forall l, \forall r, (p_l, p_r) \in \psi_m^2, sf_l(p_l, p_r) < \theta \cap sf_r(p_l, p_r) < \theta \end{aligned} \tag{3}$$

To prevent trivial solutions, the correspondence map Ψ cannot be empty and has to include all points $p_{i,j}$ from all views $\mathcal{V}$. Moreover, all points in a correspondence set Ψ_m must have a pairwise similarity below the θ threshold. In other words, two 2D points cannot be matched together if they are not similar enough. Note that a 2D point can be alone in its correspondence set (*i.e.,* $\Psi_m = \{p_{i,j}\}$). In this case, the corresponding 3D point cannot be triangulated, but it still means that it corresponds to an interesting feature of the object. Thus, correspondence sets with a single point always count towards the total count for counting applications.

2.2. Literature

Three main categories of related work deal with multi-view triangulation and sparse multi-view correspondence problems. We also review works solving the joint multi-view correspondence and triangulation problem, as well as works that focus on counting plant organs from multiple views.

Triangulation from multiple views: Triangulation from only two views can be solved by computing the pseudo intersection between the two rays. However, triangulation from multiple views is more complicated because multiple 3D lines may not intersect at the same point. A commonly used method to solve the multi-view triangulation problem is to estimate a pseudo-intersection point with the Direct Linear Transform (DLT) algorithm (Bradski, 2000; Hartley and Zisserman, 2003; Hartley and Sturm, 1997), and then use the Bundle Adjustment algorithm (Triggs et al., 1999) to refine the estimated position by minimizing the reprojection error. Multi-view triangulation has been extensively studied, and we refer the reader to the survey paper from Chen et al. (2020) for a comparison of the state-of-the-art methods for multi-view triangulation.

The dense and sparse correspondence problems: Unlike the dense correspondence problem (Dellaert, 2001; Roy and Cox, 1998), which recovers a dense depth map from two or more images, the sparse correspondence problem (Maciel and Costeira, 2003) matches a set of discrete points over multiple views. It has been extensively studied as it is a fundamental problem in computer vision, such as object tracking (Yilmaz et al., 2006), 3D reconstruction (Han et al., 2019), and image registration (Anwar et al., 2020). A well-established method for a rigid motion is to detect and describe key points with SIFT (Lowe, 2004), match them, and then use RAndom SAmple Consensus (RANSAC) (Fischler and Bolles, 1981) to discard outliers. For our problem, it is, however, not possible to assume a rigid motion between images because our plants are imaged by a single camera and are rotated after each shot. In general, if the images are not homographies but a similarity measure between points is available, solving the two-views sparse correspondence problem is possible by finding the best matching between the two sets of points. Early solutions to this problem (*e.g.,* Scott and Longuet-Higgins, 1991; Shapiro and Brady, 1992) solve the correspondence problem with more than two views, and its accuracy has been recently improved (Maset et al., 2017). However, heuristics solutions are necessary because this problem is NP-complete (Dellaert, 2001; Okutomi and Kanade, 1993).

An early solution to the sparse correspondence problem (Maciel and Costeira, 2003) expresses the correspondence problem as an integer optimization. However, the algorithm matches image sequences instead of unordered collections of images. Recent works (Bernard et al., 2019; Chen et al., 2014; Fathian et al., 2020; Huang and Guibas, 2013; Maset et al., 2017; Pachauri et al., 2013; Yu et al., 2016; Zhou et al., 2015) state the problem as finding a set of partial permutation matrices between all pairs of images in isolation while ensuring cycle consistency, which exploits the fact that the composition of pairwise matches along any loop should give the identity. These approximate methods are less memory demanding than the method from Maciel and Costeira (2003), and they scale well with the number of views. Recently, Maset et al. (2017) reported that instances with up to hundreds of points in hundred of views could be solved. The main limitation is that these methods favor precision over recall by focusing on finding consistent matches. Moreover, these algorithms require an initial configuration of the correspondences and may need parameters other than the 2D points and camera calibrations. For example, the work from Maset et al. (2017) needs an estimation of the number of points to reconstruct as an input, which makes it difficult to operate in environments with occlusions.

Most studies have relied on solving the correspondence and triangulation problems separately, making optimizing the best triangulation correspondences challenging.

Correspondence and triangulation: Cheng et al. (1994) solve the correspondence problem and triangulate two sets of points from two calibrated images. In particular, they introduce a similarity measure between two points from two different calibrated views: computing the distance between the 2D points and the reprojection of a 3D point triangulated from them. The matching between points is calculated as the maximum flow in a bipartite graph whose edges are weighted by the similarity measure. The algorithm from Cheng et al. (1994) works only with two views, and it cannot handle occlusion. This motivates our work, and we extend it to the multi-view case using a dynamic programming approach.

Bedekar and Haralick (1996) extended the work from Cheng et al. (1994) towards the multi-view case. Their brute force approach solves the correspondence problem by applying a statistical test on each possible n-tuples of 2D points. If the probability that a set of 2D points from the n-views are in correspondence is higher than a certain threshold, they are triangulated, and the resulting 3D point is kept. This algorithm is computationally demanding, and its complexity grows exponentially in the number of triangulated points. Another drawback is that occlusion is not considered. Hence, each point must be visible from all cameras.

Recently, Xiao et al. (2019) proposed a framework to calibrate a multiple-camera system (MCS) and reconstruct dynamic points for passive motion capture. They introduce an approximate algorithm to solve the correspondence problem called cyclical voting, which includes multiple pairs of global voting and in-group voting. During the triangulation step, outlier correspondences are further excluded. They evaluate instances with 200 points seen by 15 cameras. One major drawback is that their method requires the number of points to reconstruct as an input, which is an impractical assumption for counting applications. Contrary to previous work, our algorithm minimizes reprojection error by solving the joint correspondence and triangulation problem.

Plant organ counting from multiple views: Previous studies have almost exclusively focused on counting plant organs from a single view. However, some work investigated using multiple views for counting purposes to improve accuracy and increase robustness to occlusion.

Shi et al. (2019) use a full 3D reconstruction pipeline from ten views. To segment plant organs, first, each view is segmented in 2D with a fully convolutional network (FCN), then the 3D segmentation is obtained by aggregating 2D segmentations with a voting strategy. Once the 3D reconstruction pipeline has been completed, counting plant organs from the plant segmentation is straightforward. One limitation of this method is that the 3D reconstruction may fail if the plant undergoes a non-rigid movement between shots. Boogaard et al. (2020) use multi-view images to measure the internode length in cucumber plants. Nodes are detected from multiple viewpoints around the plant with a deep convolutional neural network. A clustering algorithm is used to combine the detected nodes from multiple images. One limitation of this approach is that plants are modeled as flat objects, and the clustering algorithm assumes that the distance from the camera to the plant is constant. Consequently, nodes cannot be 3D triangulated. This method is, therefore, specific to the imaging setup and the geometry of the plant. Lv et al. (2022) use multi-view images to count leaves of Physalis plants even in the presence of occlusion. First, leaves are detected in all images using a Mask R-CNN neural network. Leaves are then tracked using SIFT feature matching between successive views. One limitation of this approach is that views need to be ordered, and the tracking algorithm fails when successive views are too different from each other.

Previous methods are ad hoc and do not work in the general case. To the best of our knowledge, no method exists for counting and 3D triangulation of points of interest from multiple views in the presence of noise and occlusion.

2.3. Method

Fig. 2 shows an overview of the different steps of our method. The optimization of Ψ and thus Q in Eq. (3) is an NP-complete problem. Trying all combinations in Ψ is intractable. An intuitive greedy approximation approach would build the solution by matching successive views in some order. We could first match V_1 and V_2, then iteratively compose the solution with points from V_{i+1}. Although this approximation can give a good result, it may not converge to the optimal correspondence Ψ^*. Let us assume that two 2D points $\{p_i, p_j\}$ from two different views V_i and V_j such that $i < j$ can be matched to the same 3D point Q_n. With this greedy approach point, p_i will always be matched first to Q_n, whether or not it is the optimal assignment. Once this suboptimal decision is made, we cannot backtrack. Therefore, we may try all possible orders of views and keep the ordering that minimizes the re-projection error $E(\Phi)$ from Eq. (2).

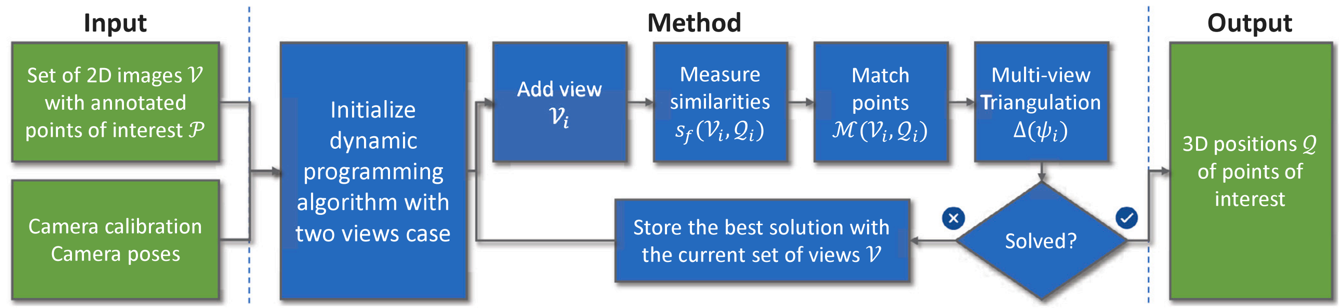

Input
Set of 2D images $\mathcal{V}$ with annotated points of interest $\mathcal{P}$
Camera calibration
Camera poses
Method
Initialize dynamic programming algorithm with two views case
Add view $\mathcal{V}_i$
Measure similarities $s_f(\mathcal{V}_i, \mathcal{Q}_i)$
Match points $\mathcal{M}(\mathcal{V}_i, \mathcal{Q}_i)$
Multi-view Triangulation $\Delta(\psi_i)$
Solved?
Store the best solution with the current set of views $\mathcal{V}$
Output
3D positions $\mathcal{Q}$ of points of interest

Fig. 2. Method overview: Green boxes represent input and output data, blue boxes represent processing steps of our method. For clarity, this figure considers the dynamic programming algorithm as a bottom-up approach. First, the algorithm is initialized with all pairs of views. Then, views are iteratively added to all candidate solutions. The process of adding a view to a candidate solution is described in Alg. 1 and consists in (1) measuring similarities between 2D points $\mathcal{P}_i$ in the view $\mathcal{V}_i$ and candidate 3D points $\mathcal{Q}$, (2) matching 2D points $\mathcal{P}_i$ to candidate 3D points $\mathcal{Q}$, and (3) triangulating the resulting 3D points. When all views have been added to all candidate solutions, the solution that minimizes the energy $\mathcal{E}(\psi)$ is selected. This iterative process is repeated until all views have been added to the problem, and the solution is found.

2.3.1. Dynamic programming

Let us assume that the problem is solved for k views. If we remove one view, the solution to the multi-view correspondence problem with the remaining $k-1$ views will essentially be the same as when solving the problem directly with $k-1$ views. There is no interdependence between views. This is the optimal substructure of our problem, suitable for a dynamic programming approach. Another way of interpreting our dynamic programming approach is that it efficiently tries all orders of views and always keeps the order that best minimizes the re-projection error $E(\Phi)$ from Eq. (2).

We denote $\{\mathcal{Q}, \Psi\} = \mathcal{S}(\mathcal{V})$ the procedure that solves the joint problem with views $\mathcal{V}$ and outputs the triangulated points $\mathcal{Q}$ and the correspondences Ψ. The recurrence relation that is the key to the dynamic programming approach is:

$$\mathcal{S}(\mathcal{V}) = \underset{\forall V_i \in \mathcal{V}, \mathcal{Q}, \Psi}{\arg\min} \quad E(\Psi) \qquad (4)$$
$$\textit{subject to } \{\mathcal{Q}, \Psi\} = \mathcal{M}(V_i, \mathcal{S}(\mathcal{V} \setminus V_i)).$$

Function $\mathcal{S}(\mathcal{V})$ solves the problem for a set of k views $\mathcal{V}$ (detailed in Algorithm 1) as follows:

1. The problem is subdivided into sub-problems, each with $k-1$ views. Each view $V_i \in \mathcal{V}$ is removed from the set of views $\mathcal{V}$ and we solve the sub-problem with $\{\mathcal{V} \setminus V_i\}$.
2. We match the 2D points from V_i with the result of the sub-problem using the matching function $\mathcal{M}$ (Section 2.5) that returns a candidate set of correspondences Ψ, taking into account points from view V_i.
3. We triangulate and evaluate the re-projection error with $E(\Psi)$.
4. Only the solution of the sub-problem that minimizes the re-projection error $E(\Psi)$ is kept. The procedure $\mathcal{S}$ is called recursively in a dynamic programming manner. We use memorization *i.e.,* the results of the overlapping sub-problems are stored in a lookup table $\mathcal{C}$. This way, we solve a total of 2^k sub-problems instead of $k!$.

Algorithm 1: Dynamic Programming solution to the problem: $\{\mathcal{Q}, \Psi\} = \mathcal{S}(\mathcal{V})$

```
Input   : A set of views $\mathcal{V}$
Data: 2D points $\{\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_k\}$ related to $\mathcal{V}$, the memoization lookup table $\mathcal{C}$
Output  : The triangulated 3D points $\mathcal{Q}$ and correspondences $\Psi$
// Search for a cached result in the lookup table
if $\mathcal{V} \in \mathcal{C}$ then
    // Return the cached result
    return $\mathcal{C}[\mathcal{V}]$;
// Initialize the best configuration
$minError \leftarrow \infty$;
$min\mathcal{Q} \leftarrow \{\}$;
$min\Psi \leftarrow \{\}$;
for $V_i \in \mathcal{V}$ do
    // Solve the sub-problem recursively without view $V_i$
    $\{\mathcal{Q}', \Psi'\} \leftarrow \mathcal{S}(\mathcal{V} \setminus V_i)$;
    // Match the 2D points from $V_i$ on the result of the sub-problem
    $\Psi \leftarrow \mathcal{M}(P_i, \{\mathcal{Q}', \Psi'\})$;
    // Triangulate the new correspondences
    $\mathcal{Q} \leftarrow \Delta(\Psi)$;
    // Evaluate the re-projection error of this candidate solution
    $error \leftarrow E(\Psi)$;
    // Keep only the solution that minimizes $E$
    if $error < minError$ then
        $minError \leftarrow error$;
        $min\mathcal{Q} \leftarrow \mathcal{Q}$;
        $min\Psi \leftarrow \Psi$;
// Store the result in the lookup table
$\mathcal{C}[\mathcal{V}] \leftarrow \{min\mathcal{Q}, min\Psi\}$;
return $\{min\mathcal{Q}, min\Psi\}$;
```

2.4. Similarity function

The similarity function is denoted by $sf_b(p_l, p_r)$, and it evaluates the similarity of two 2D points p_l and p_r from different calibrated views V_l and V_r. It is inspired by the similarity function from Cheng et al. (1994, Sec 2). There exist two rays r_l and r_r that start from the two camera centers and go in the direction of p_l and p_r respectively. Note that if the approximate geometry of the scene is known, rays can be clamped to line segments to improve the robustness of the similarity measure *i.e.,* rays that intersect at infinity will not be considered similar. We compute the optimal 3D pseudo-intersection M between the two rays. From there, 3 cases can happen: (1) r_l and r_r are parallel ($sf_b(p_l, p_r) = \infty$). (2) The pseudo-intersection M is in the negative direction on one of the two rays ($sf_b(p_l, p_r) = \infty$). (3) M is within the 3D scene and is visible from both views. In this case, we project M on both views and get: the 2D point M_l on view V_l, and the 2D point M_r on view V_r. The similarity function has two terms, one for each view:

$$\begin{aligned} sf_l(p_l, p_r) &= \left\| M_l - p_l \right\|_2 \qquad (5) \\ sf_r(p_l, p_r) &= \left\| M_r - p_r \right\|_2 \\ sf_b(p_l, p_r) &= sf_l(p_l, p_r) + sf_r(p_l, p_r) \end{aligned}$$

If $sf_b(p_l, p_r) = \infty$, the two 2D points are dissimilar. If $sf_b(p_l, p_r) = 0$, the two 2D points are perfectly similar and most likely, they are the projection of the same 3D point. In addition, the hyper-parameter θ defines the threshold in pixels beyond which two points are not similar enough to be matched. In other words, if $sf_l(p_l, p_r) < \theta \cap sf_r(p_l, p_r) < \theta$, p_l and p_r can be matched together, otherwise the match is prevented because p_l and p_r are too dissimilar.

We also introduce another similarity function used when matching a 2D point p_i from a calibrated view V_i to a 3D point q. In this case, we project q on the view V_i, and the value of the similarity function is the re-projection error between $P_i \cdot q$ and p_i:

$$sf_p(p_i, q) = \left\| P_i \cdot q - p_i \right\|_2 . \tag{6}$$

Computing the similarity between points via a 3D pseudo intersection is equivalent to using epipolar geometry. With epipolar geometry, we would project the ray associated to a point p_i from V_i on another view V_j and compute the orthogonal distance from the projected ray to a 2D point p_j from V_j, which is twice the re-projection error of the pseudo-intersection point on view V_j. The difference is that the similarity function computes the re-projection error on both views, which is the same as the re-projection error when triangulating.

2.5. *Matching function*

The function $\mathcal{M}()$ matches 2D points from calibrated views. There are two cases: (1) matching two views, and (2) adding points from a new view to a solved sub-problem.

Matching two views: V_l and V_r bootstraps the dynamic programming algorithm, and we use the algorithm from Cheng et al. (1994). Values of the bilateral similarity function $sf_b(p_l, p_r)$ between all 2D points $p_l \in \mathcal{P}_l$ and $p_r \in \mathcal{P}_r$ are stored in a matrix, which is the input to the assignment problem. To solve the assignment problem, instead of computing the maximum flow in a weighted bipartite graph like the work from Cheng et al. (1994), we use the Hungarian algorithm (Edmonds and Karp, 1972; Kuhn, 1955), which is equivalent. The output of the assignment problem is a set of correspondences Ψ that maximizes similarity *i.e.,* it finds the set of correspondences that minimizes the re-projection error from both views. In the two-views case, we use the bilateral similarity function $sf_b(p_l, p_r)$ to account for the re-projection error in both views simultaneously. Note that if two points have respective similarities greater than θ *i.e.*, $sf_l(p_l, p_r) \geq \theta$ or $sf_r(p_l, p_r) \geq \theta$, they cannot be matched together. As a consequence, if a point p_l cannot be matched to any other point p_r from the other view V_r, then it is kept in its own set of correspondences $\psi_n = \{p\}$, and will likely be matched later with a point from another view.

Adding a new view: When matching 2D points $\mathcal{P}_i$ from a view V_i to the result $\{\mathcal{Q}', \Psi'\}$ of a sub-problem $\mathcal{S}(\mathcal{V} \setminus V_i)$, for all pairs of 2D points p_i and for all correspondence sets ψ'_n, we triangulate the result of assigning p_i to ψ'_n: $Q_n = \Delta(\{\psi'_n \cap p_i\})$, then the resulting point Q_n is used to compute the similarity $sf_p(p_i, Q_n)$. If a 3D point cannot be triangulated because only one 2D point p_q is associated to it, we match $p_i \in \mathcal{P}_i$ with the 2D point p_q instead, using the unilateral similarity function $sf_l(p_i, p_q)$. All similarities are stored in a matrix used to solve the assignment problem between 2D points in view V_i and 3D points that result from the sub-problem. The Hungarian algorithm finds the set of correspondences that minimizes the change in re-projection error caused by adding the view V_i. In the presence of occlusion, the view V_i may have a different number of points than in $\mathcal{Q}'$ *i.e.*, $n_i \neq n'$, which may leave some points unassigned. Also, the view V_i may have the correct number of points, but they do not necessarily all correspond to the 3D points in $\mathcal{Q}'$. This case is handled using the threshold hyper-parameter θ (in pixels). If $sf_p(p_i, Q_n) \geq \theta$ the assignment between p_i and Q_n is prevented. After the assignment problem is solved, if a point p_i is left unassigned, it is kept in its own set of correspondences $\psi_m = \{p_i\}$. Obviously, ψ_m itself cannot produce a 3D triangulated point Q_m because it corresponds to only one 2D point, but this point will likely be assigned later to other points in other views. The optimal value for the hyper-parameter θ depends on the amount of noise feature points undergo during image acquisition. In Section 3, we show different ways in which θ can be estimated to give the best results.

A failure can be caused by noise leading to a 2D point being incorrectly matched to a 2D point that is not the projection of the same 3D point. This ambiguity is difficult to avoid when solving the correspondence problem based on the scene geometry (Cheng et al., 1994). Adding more views helps clarify the ambiguities, and our approach implicitly mitigates this problem because another solution path in the dynamic programming approach may lead to the correct matching.

Our solution always builds the set of correspondences Ψ based on the triangulated 3D points $\mathcal{Q}$. In other words, we solve the joint problem because the triangulation solution is used to improve correspondence and vice versa.

2.6. *3D triangulation*

The triangulation function $\Delta(\Psi)$ uses the Direct Linear Transform algorithm (Hartley and Sturm, 1997) to estimate the pseudo-intersection point, and refinement with Bundle adjustment (Triggs et al., 1999) that minimizes the re-projection error. Note that this method does not guarantee that the global optimum is found. The consequence is that the global minimum of Eq. (3) cannot be reached. In our case, even though the multi-view triangulation function does not find the global optimum, experimentally, we show that it finds a solution close enough to optimal such that it does not impair the quality of the solution to the correspondence problem.

2.7. *Computational complexity*

For the algorithm's worst-case complexity, we deliberately ignore occlusion and noise because that maximizes the number of operations when matching 2D points from views.

When triangulating a 3D point seen from k views, the DLT algorithm solves a linear system with an SVD decomposition that depends on the number of views k. This operation's complexity is at most $\mathcal{O}(k^3)$. After DLT, Bundle Adjustment runs a limited number of iterations (approx. 6), each of which takes $\mathcal{O}(k)$. Therefore, the triangulation is done in $\mathcal{O}(k^3)$.

The input to our algorithm is a collection of n 3D points seen from k calibrated views $\mathcal{V}$. For each problem of the dynamic programming approach, we match at most k views with n points with the results of sub-problems. The matching is done using the Hungarian algorithm, that requires a matrix of similarities with n^2 coefficients, each of which requires (1) the triangulation of a 3D point seen from k views in $\mathcal{O}(k^3)$, and (2) the computation of the similarity coefficient in $\mathcal{O}(1)$. In total, all similarities are computed in $\mathcal{O}(n^2 \cdot k^3)$. The assignment problem is solved using the Hungarian algorithm in $\mathcal{O}(n^3)$. Regarding the last multi-view triangulation, n points are triangulated in $\mathcal{O}(n \cdot k^3)$. Since there are at most k sub-problems, the total complexity of solving a problem is: $\mathcal{O}\left(k \cdot (n^2 k^3 + n^3)\right)$. Asymptotically, the complexity per problem is dominated by the matching part because the number of views is small compared to the number of points: $k \ll n$. Therefore, the time complexity per problem is $\mathcal{O}(k \cdot n^3)$. When solving sub-problems with dynamic programming, we solve all subsets of $2, 3, \ldots, k$ views, which gives a total of 2^k sub-problems solved. In conclusion, the complexity is $\mathcal{O}(k \cdot n^3 \cdot 2^k)$.

2.8. *Implementation*

Our method was implemented in C++ with OpenCV (Bradski, 2000). We ran it on a PC workstation equipped with an Intel Xeon W-2145 running at 4.5 GHz. We did not parallelize our code, nor did we used GPU acceleration. We developed a Python interface to the C++ code to make it easier for other researchers to re-use our algorithm. The reference implementation will be available on GitHub upon acceptance.

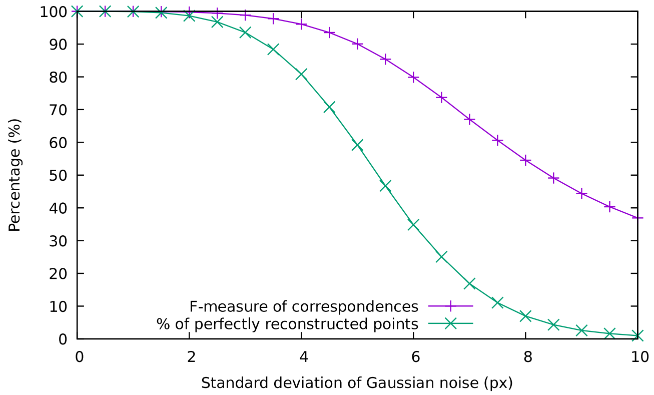


Fig. 3. Evaluation with varying noise levels: ten points were generated and projected to six cameras without occlusion. The standard deviation of the Gaussian noise added to all 2D points varies between 0–10 pixels. We generated 10,000 different configurations and computed (1) the average F-measure and (2) the percentage of perfectly reconstructed points. Some points might be only partially detected (detected in only some views), which explains why the F-measure is higher than the percentage of perfect points.

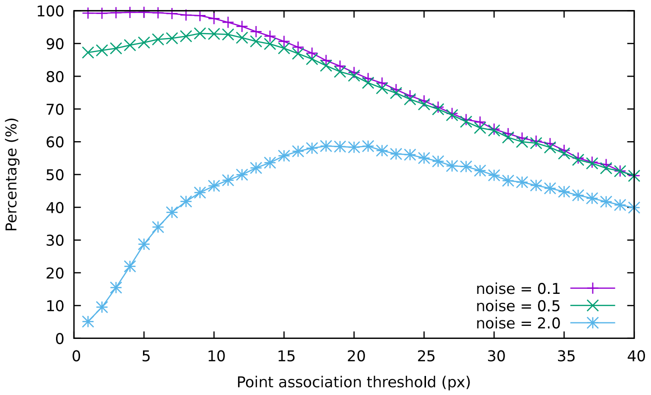


Fig. 4. Occlusion: Evaluation with varying threshold θ for pairing points between views. A total of 20 points were generated and projected to six cameras with 50% occlusion. The standard deviation of the Gaussian noise added to all 2D points was: 0.1, 0.5, or 2.0 pixels. We generated 10,000 different configurations and computed the percentage of perfectly reconstructed points.

3. Validation

Three different experiments were conducted to evaluate our algorithm: (1) a synthetic data set is generated and used as ground truth to validate the correspondences and triangulated points found by our algorithm. Additional measures are computed to assess the accuracy and robustness of our algorithm in the presence of noise and occlusion (Section 3.1). We also show a real example: (2) A phenotyping use case with a set of 34 sorghum plants imaged from 6–10 points of view (Section 3.2). Experts were asked to annotate the position of leaf tips in all views. Our algorithm is run on the dataset to count the number of leaves per plant using the information from all views; results are compared to manual annotations made using observations by experts who observed and counted the leaves of real plants, rather than estimating leaf number from 2D images. In addition, we show in the appendix a synthetic sorghum dataset. This experiment gives a theoretical upper bound on the accuracy of our algorithm by simulating a perfect image acquisition. Finally, we show in appendix (3) an evaluation of the precision and accuracy of the full measurement pipeline: a LEGO brick is measured by labeling its corners in five calibrated views (Appendix A).

3.1. Synthetic benchmark

We generated n 3D points on a sphere of diameter 1 m and k cameras on a concentric sphere of 3 m diameter. Cameras were oriented toward the 3D points, and the projection of points is simulated on cameras with a resolution of $1,000 \times 1,000$ pixels. After the points were projected, we added Gaussian noise to their 2D projections with zero mean and varying standard deviation (between zero and ten pixels). To simulate occlusion, random 2D points were deleted with a uniform probability. We only keep a valid configuration that can be reconstructed, *i.e.,* each 3D point must be visible from at least two views. However, it is possible that none of the views can see all n 3D points. We keep track of the ground-truth position of 3D points and correspondences between the views.

Evaluation of the triangulated points Q: The resulting points from the triangulation match the ground-truth 3D points. A *misdetected point* is a point that is missing in the triangulated set but is in the set of ground-truth 3D points. A *false alarm point* has been triangulated but is not in the set of ground-truth 3D points. The *reconstructed points* are detected by our algorithm and present in the ground-truth set. If a reconstructed point has precisely the same correspondences as its ground-truth counterpart (F-measure = 1), we call it a *perfectly reconstructed point.*

Evaluation of the correspondences Ψ: The resulting correspondences are also matched with the ground truth. A *true positive* tp is a correspondence between a 3D point and a 2D point present in both the result and the ground truth. A *false positive* fp is a correspondence present in the result but not in the ground truth. Conversely, a *false negative* fn is a correspondence that is not present in the result even though it is part of the ground truth. To aggregate these statistics, we compute the precision $p = tp/(tp + fp)$, recall $r = tp/(tp + fn)$ and F-measure $f = 2 \cdot (p \cdot r)/(p + r)$. F-measure values $f \approx 1$ mean that the correspondence problem has been successfully solved (*perfectly reconstructed points*) and $f \approx 0$ means that the correspondence failed. Finally, to evaluate the triangulation quality, we measure the distribution of distances between the triangulated 3D points and their ground-truth equivalents. We measure the minimum, first quartile, median, third quartile, maximum, mean, and standard deviation.

Noise: Fig. 3 shows the evaluation of the solution to the correspondence problem with varying levels of noise. The correspondence shows that over 80% of 3D points are perfectly reconstructed until the noise reaches a standard deviation above four pixels *i.e.,* when nearly all 2D points are within a disk with a radius of 12 pixels (about 2% relative error w.r.t. the image resolution), the solution is accurate.

Occlusion: Fig. 4 helps determine the best value of the threshold θ with respect to a fixed noise level in the presence of 50% occlusion. About half of the points were occluded *i.e.,* on average each view can see only 10 out of the 20 3D points. Reciprocally, on average, a 3D point is visible from three views. We aim to find the threshold that maximizes the F-measure of correspondences for a certain level of noise and occlusion. For $noise = 0.1$, the best threshold is eight pixels. For $noise = 0.5$, the best threshold is 11 pixels. For $noise = 2.0$, the best threshold is 22 pixels. The trend is that when noise increases, the value of the optimal threshold θ increases as well.

Triangulation: Fig. 5 shows that the distance from the triangulated points to the ground-truth 3D points reduces when adding views. This result is in accordance with Bedekar and Haralick (1996). For example, with the noise set to a standard deviation of 2 px, the median distance decreases from 22.8 mm with two cameras to 7.8 mm with ten cameras. When adding views, the gain in accuracy plateaus out. Note that, with only two cameras, the mean distance is higher than the median because some points are not perfectly corresponded due to noise.

Runtime: Fig. 6 shows the average runtime of our algorithm with a varying number of points as well as the polynomial trend line for comparison.

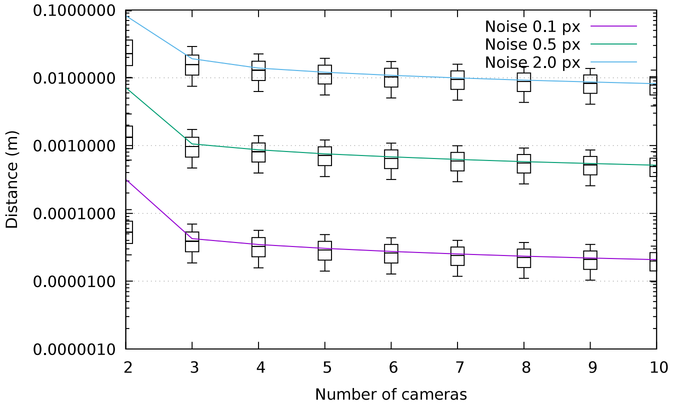


Fig. 5. Triangulation: Mean (curve) and box plots (first and last deciles, quartiles, and median) of the distance from triangulated points to ground-truth 3D points with a varying number of cameras. Ten points were generated and projected on 2–10 cameras without occlusion. We generate 10,000 different configurations and compute the first decile, first quartile, median, third quartile, last decile, and mean. We show the evolution of the triangulation accuracy for three different values of the standard deviation of noise: 0.1 px, 0.5 px, and 2 px.

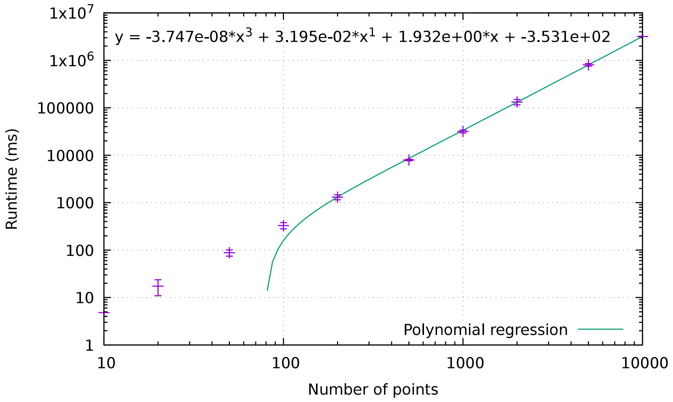


Fig. 6. Average runtime in milliseconds (points) and standard deviation (error bars) of our algorithm with a varying number of points. Between 10 and 10,000 points were generated and projected on three cameras without noise or occlusion.

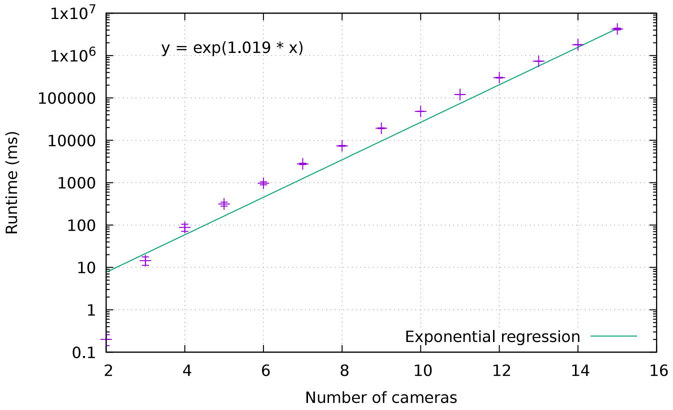


Fig. 7. Average runtime in milliseconds (points) and standard deviation (error bars) of our algorithm with a varying number of cameras. A total of 20 points were generated and projected on 2–15 cameras without noise or occlusion.

Fig. 7 shows the average runtime of our algorithm with a varying number of cameras. We can see that the runtime is exponential with regard to the number of cameras.

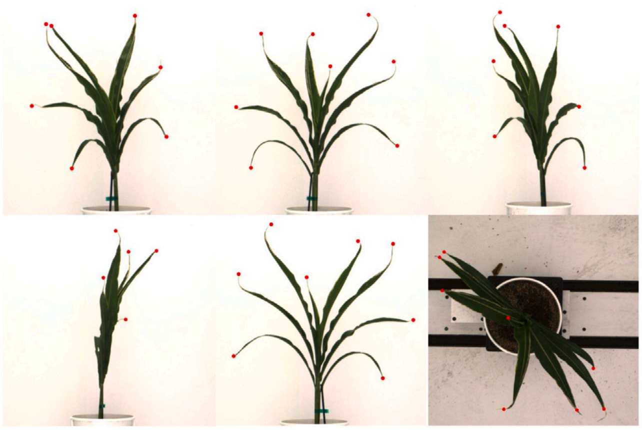


Fig. 8. Annotation of leaf tips (red points) on the six views of a sorghum plant from Gaillard et al., 2020. Experts annotated the leaf tips in each of the six images, on which 7, 8, 7, 5, 8, and 8 tips are visible. Triangulations of leaf tips are in Fig. 9.

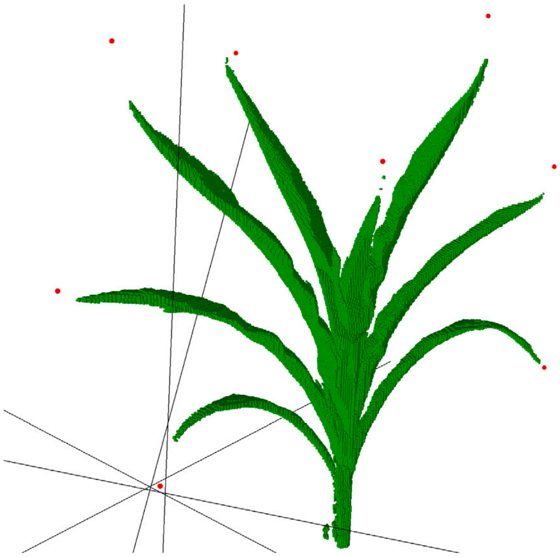


Fig. 9. Triangulation of leaf tips of a sorghum plant displayed along with its voxel 3D reconstruction (from Gaillard et al., 2020). Red points show the eight triangulated leaf tips annotated from six plant views. The five rays that correspond to the bottom left tip are shown in black.

3.2. *Leaf counting*

This section shows an application of our algorithm to enhance the accuracy of leaf counting from multi-view images. In particular, we are counting leaves from sorghum plants imaged at a phenotyping facility using a turn-table imaging system. Our algorithm identifies correspondences between views by reasoning on the 3D geometry of the scene. In contrast, the current state-of-the-art in sorghum leaf counting (Miao et al., 2021) identifies the view in which the plant has the widest projection (the plant is facing the camera), and automatically detects leaf tips, only in this view. As a reference, we simulate this previous technique by taking the maximum number of leaf tips visible in any of the views; this is referred to as the *baseline* throughout the rest of this section.

To show the versatility of our approach, two different phenotyping setups were used for validation. The first dataset was collected using the original RGB cameras installed at the University of Nebraska phenotyping facility with a single top picture and five side view pictures collected at angles 0, 72, 144, 216, 288 degrees, with a resolution of

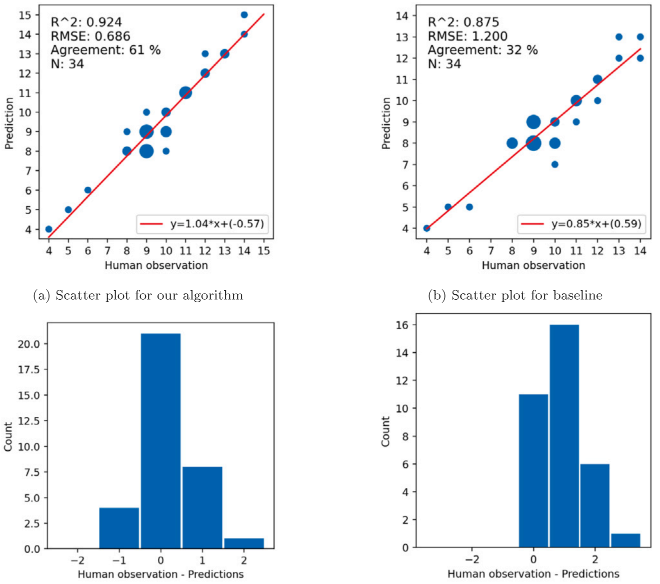
Table 1
Evaluation of the robustness of our algorithm in comparison to the baseline. To simulate detection 1) false positives, annotations were added with a probability between 0% and 10%, and 2) false negatives, annotations were discarded with a probability between 0% and 50%. The table shows the average agreement and average RMSE w.r.t. the percentage of discarded points among five runs with different random seeds. Cells with bold font correspond to the best agreement or RMSE between our method and the baseline. See Fig. 11 for graphs corresponding to the data in this table.

Dataset	[%]	Our method		Baseline	
		Agreement	RMSE	Agreement	RMSE
Sorghum 2022	−10%	**8.80 ± 8.50**	3.01 ± 0.29	5.60 ± 3.13	**1.99 ± 0.18**
	−8%	**10.20 ± 4.38**	2.45 ± 0.46	7.40 ± 7.50	**1.63 ± 0.32**
	−6%	**25.80 ± 16.08**	1.68 ± 0.43	21.20 ± 11.50	**1.31 ± 0.27**
	−4%	**35.00 ± 13.00**	**1.07 ± 0.34**	31.80 ± 19.92	1.16 ± 0.36
	−2%	**67.20 ± 8.50**	0.81 ± 0.13	59.60 ± 16.56	**0.72 ± 0.13**
	0%	**100.00 ± 0.00**	**0.00 ± 0.00**	92.00 ± 0.00	0.28 ± 0.00
	5%	**95.20 ± 4.38**	**0.17 ± 0.15**	92.00 ± 0.00	0.28 ± 0.00
	10%	**95.20 ± 4.38**	**0.17 ± 0.15**	84.00 ± 5.66	0.39 ± 0.07
	15%	**92.00 ± 5.66**	**0.24 ± 0.15**	69.00 ± 16.26	0.54 ± 0.16
	20%	**79.40 ± 8.82**	**0.48 ± 0.13**	56.40 ± 15.82	0.78 ± 0.24
	25%	**64.20 ± 4.38**	**0.73 ± 0.07**	33.40 ± 11.74	1.13 ± 0.23
	30%	**52.00 ± 12.63**	**0.90 ± 0.16**	16.40 ± 12.72	1.39 ± 0.28
	35%	**36.60 ± 13.92**	**1.15 ± 0.18**	11.80 ± 7.16	1.70 ± 0.29
	40%	**28.80 ± 9.93**	**1.33 ± 0.23**	8.80 ± 8.50	2.08 ± 0.37
	45%	**10.20 ± 4.38**	**1.65 ± 0.24**	5.60 ± 3.13	2.51 ± 0.33
	50%	**5.80 ± 6.22**	**2.01 ± 0.22**	4.20 ± 3.83	2.90 ± 0.37
Sorghum 2018	−10%	2.60 ± 3.97	3.11 ± 0.41	**10.00 ± 4.18**	**1.94 ± 0.24**
	−8%	9.80 ± 8.56	2.58 ± 0.28	**15.80 ± 5.36**	**1.64 ± 0.33**
	−6%	10.00 ± 8.97	2.14 ± 0.31	**16.60 ± 6.19**	**1.38 ± 0.23**
	−4%	26.80 ± 8.50	1.61 ± 0.17	**31.00 ± 10.27**	**1.04 ± 0.17**
	−2%	50.00 ± 11.42	1.02 ± 0.12	**65.00 ± 4.18**	**0.63 ± 0.04**
	0%	**90.00 ± 0.00**	**0.49 ± 0.00**	**90.00 ± 0.00**	**0.49 ± 0.00**
	5%	**74.00 ± 4.47**	**0.64 ± 0.23**	68.00 ± 5.70	0.68 ± 0.04
	10%	**70.00 ± 10.75**	**0.64 ± 0.24**	50.80 ± 6.10	0.97 ± 0.15
	15%	**59.60 ± 11.55**	**0.80 ± 0.19**	35.60 ± 6.19	1.32 ± 0.10
	20%	**57.60 ± 10.31**	**0.90 ± 0.13**	24.40 ± 6.07	1.68 ± 0.12
	25%	**44.20 ± 8.76**	**1.10 ± 0.13**	12.80 ± 7.05	2.01 ± 0.09
	30%	**35.40 ± 9.21**	**1.22 ± 0.13**	5.20 ± 3.83	2.41 ± 0.12
	35%	**24.20 ± 3.83**	**1.54 ± 0.09**	3.40 ± 3.71	2.87 ± 0.11
	40%	**16.60 ± 6.19**	**1.86 ± 0.12**	2.40 ± 2.19	3.43 ± 0.16
	45%	**8.00 ± 5.48**	**2.25 ± 0.17**	1.60 ± 2.19	3.90 ± 0.18
	50%	**8.00 ± 4.18**	**2.74 ± 0.14**	1.60 ± 2.19	4.41 ± 0.09

2, 454 × 2, 056 px (Ge et al., 2016). A set of 21 sorghum plants, initially imaged in 2018 and previously described (see Gaillard et al., 2020 for details) were employed as part of this study. The second dataset was collected using upgraded cameras and new imaging protocols put in place in 2020. Each plant is imaged with a top picture, and ten side pictures, taken at angles 0, 36, 72, 108, 144, 216, 252, 288, 384 degrees, with a resolution of 4, 384 × 6, 576 px. For this data set, 13 sorghum plants were imaged in 2022. For all plants, experts counted the number of leaves on the live plant and annotated leaf tips in all views. For example, Fig. 8 shows an annotated plant from the 2018 dataset, and Fig. 9 shows the result of the triangulation for this plant. Overall, the 34 plants amount to a total of 256 images (21 plants with six views and 13 plants with ten views) annotated by humans. Particular attention was paid to the quality of the annotation process. In ambiguous cases, additional expert annotators were consulted to generate a consensus annotation. Plants, where no consensus could be reached, were dropped from consideration. Therefore, we can consider these human annotations to be of high quality and reflect the ground truth. This is important and allows us to evaluate the performance of our algorithm in the best conditions. Starting from these ground-truth annotations, we simulate adversarial cases by adding fake annotations (false positives) or removing annotations (false negatives).

To evaluate our algorithm on real data, we run two experiments with our sorghum datasets.

Leaf counting: Given the human annotations in all views, we triangulate and count leaf tips in all 34 plants. Our algorithm aggregates the annotations from multiple views, whereas the baseline simulates single-view methods by taking the maximum number of annotations seen in any view. The prediction results are compared to the ground-truth leaf numbers.



(a) Scatter plot for our algorithm

(b) Scatter plot for baseline

(c) Histogram for our algorithm

(d) Histogram for baseline

Fig. 10. Relationship between predicted leaf number and human annotated leaf number for our algorithm and the baseline. In the top row, subfigures (a) and (b) show the scatter plot of predicted leaf number versus human annotated leaf number. In the bottom row, subfigures (c) and (d) show the histogram of the distribution of errors (difference between the human-annotated leaf number and the predicted leaf number). In the left-hand column, subfigures (a) and (c) correspond to the results of our algorithm (aggregate of multiple views), and in the right-hand column, subfigures (b), and (d) correspond to the results of the baseline (maximum number of annotation among all views). Note that the baseline corresponds to the best-case scenario for any single-view leaf counting method: detection neural network or human annotator *i.e.,* no false negative or false positives. In the top, left corners of the scatter plots: R^2 is the square of the correlation coefficient. RMSE is the root of the mean squared error. The Agreement rate is the percentage of perfect predictions. N is the number of plants in the data set. In scatter plots, the linear regression line is in red, and the corresponding equation is written in the bottom right corner.

Fig. 10 shows scatter plots and histograms that compare our algorithm to the baseline. The scatter plots show the predicted number of leaves versus the human-annotated leaf number. We compute additional metrics: linear regression with the correlation coefficient R^2, the Root Mean Squared Error (RMSE), and the Agreement, which is the percentage of plants whose number of leaves is perfectly predicted.

The histograms show the prediction error, that is, the difference between the human-annotated leaf number and the predicted leaf number. For example, an error of −2 means that two extra leaves are predicted, and an error of 1 indicates that one leaf is missing in the prediction. For this experiment, $\theta = 200$ px for the sorghum 2018 dataset, and $\theta = 750$ px for the sorghum 2022 dataset. The procedure to determine these values of θ is described in the robustness experiment (see the second next paragraph). We can see that starting from ground-truth annotations, our algorithm could successfully count leaves in 1 more plant than the baseline, which is reflected in the agreement (an increase from 91% to 94%).

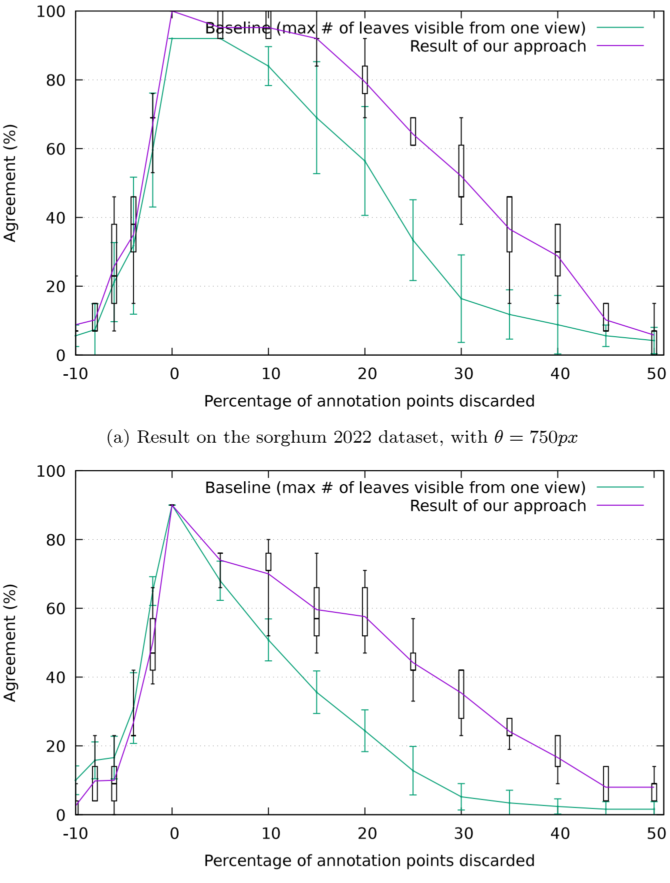
Fig. A.15 in the appendix shows the same leaf counting experiment as in Fig. 10 with 20% of annotations discarded. This figure shows the robustness of our algorithm for leaf counting compared to the baseline when some of the annotations are missing. In particular it shows that even when the agreement is not close to 100%, predictions that are not exact are still close to the ground truth and the spread of the distribution of errors is narrow ($RMSE = 0.686$).

Robustness to occlusion: To evaluate the robustness of our algorithm we simulate detection (1) false positives by adding random annotations with a certain probability, and (2) false negatives by discarding annotations with a certain probability. The probability of adding annotations ranges from 0% to 10% in increments of 2%, and the probability of discarding annotations is varying between 0% and 50%, in increments of 5%. For each probability of adding or discarding annotations, we measure the average agreement over 5 runs with different random seeds. This experiment shows that our algorithm can improve the counting accuracy even when annotations are not of perfect quality. To account for the simulated occlusion, the θ hyper-parameter needs to be set to a value below infinity, otherwise the matching function will enforce matching of points that are not necessarily similar. To find the best value of θ we ran the evaluation with different thresholds: $\theta \in \{200, 500, 600, 750\}$ and selected the best. For the sorghum 2018 dataset, the best value is 200 px. For the sorghum 2022 dataset, the best value is 750 px. Fig. A.14 in the appendix shows the plot of the average agreement for different values of θ.

Fig. 11 and Table 1 show the average agreement w.r.t. the percentage of added or discarded annotations for the two different datasets. In Fig. 11, for the baseline, the curves show the mean agreement and the standard deviation. For our approach, the curves show the mean agreement, and for each measurement a box plot with the minimum, first-quartile, median, last-quartile, and maximum is also shown. Having a sense of the distribution of agreements among all runs allows us to better compare our algorithm to the baseline. We can see that our algorithm consistently give results with a better average agreement than the baseline. Moreover, the minimum agreement found by our approach is often greater than the average plus the standard deviation of the baseline agreement, which indicates that our method is significantly better than the baseline.

Of particular interest is the agreement on the sorghum 2022 dataset when no annotations are changed. Our algorithm outperforms the baseline and gives a 100% correct result, this happens because one of the plant in the dataset has at least one leaf occluded in all views, and a single-view detection algorithm cannot aggregate results from multiple views.

Fig. A.13 in the appendix shows a synthetic benchmark that gives the theoretical best result achievable for both imaging setups of the University of Nebraska-Lincoln. It features a synthetic dataset of 100 plants with 5 to 15 leaves imaged with the simulated imaging setups. Because it has a perfect calibration, no occlusion, and no noise, this benchmark is an upper bound for the results of both the sorghum 2018 and 2022 datasets. In other words it gives the theoretical best performance of our algorithm.



(a) Result on the sorghum 2022 dataset, with $\theta = 750px$

(b) Result on the sorghum 2018 dataset, with $\theta = 200px$

Fig. 11. Evaluation of the robustness of our algorithm in comparison to the baseline. To simulate detection (1) false positives, annotations were added with a probability between 0% and 10%, and (2) false negatives, annotations were discarded with a probability between 0% and 50%. Curves show the average agreement w.r.t. the percentage of discarded points among fives runs with different random seeds. Note that a negative number of discarded points means that points were added. The green curves correspond to the baseline, and the error bars show the standard deviation. The purple curves correspond to the results of our approach. The box plots show the distribution of the results (min, first quartile, median, last quartile, max). We can see that our approach is consistently better at predicting the number of leaves than the baseline.

4. Discussion

We conducted experiments showing that our algorithm improves the leaf counting accuracy by leveraging the information from multiple views instead of relying on a single view (see Section 3.2). Our multi-view method outperforms methods based on single views in leaf counting applications, even in the presence of occlusion and noise. For example, our sorghum 2022 dataset includes a plant with ten leaves captured from ten views but a maximum of only nine leaves visible in any single view as a result of occlusion. Our algorithm successfully estimated its leaf count (see Fig. 11).

To compare with the state-of-the-art single-view approach, we give the absolute best-case scenario in which we carefully checked that annotations were perfect (no false positives, no false negatives). While the best automatic detection algorithms do not provide this performance, we wanted to make a fair comparison when none of the annotations are modified. This makes our evaluation future-proof to forthcoming advances in the field of automatic object detection. It should be noted that compared to the state-of-the-art single-view approaches, our algorithm will tend to exhibit the greatest increase in accuracy on sorghum plants with more complex canopy architectures that diverge from the idealized 2D architecture produced by perfectly alternating phyllotaxy.

We cannot prove the convergence of our algorithm formally because the multi-view triangulation step does not always reach a global optimum. We experimentally evaluate the algorithm with a synthetic benchmark and a leaf counting experiment using real plants annotated by experts, showing a very good real-world application performance.

It should be noted that both the baseline and our algorithm are more robust to false negatives than to false positives. For instance, 10% false positives lead to a similar loss in agreement as with 50% false negatives. The baseline takes the maximum of annotations in all views, which is not as robust to annotations added as to annotations discarded. Similarly, our algorithm is designed to be robust to occlusion, but not to false positives, because it trusts that any annotation (good or bad) corresponds to an object of interest to count. To mitigate this issue, the solution is to tune the automatic detection neural network to favor precision over recall, because a leaf tip not detected in a view may be detected in other views.

Fig. 11 shows that our algorithm performs better on the sorghum 2022 dataset than on the sorghum 2018 dataset. We did not have a precise estimation of the camera calibration parameters for the 2018 imaging setup. Moreover, the imaging setup in the 2022 dataset is of better quality, with better optics and a camera of higher resolution.

These upgrades result in less noise per image. In addition, improvements to the throughput of the imaging protocol allowed more views per plant to be acquired in 2022 relative to 2018, further improving results. Note that we excluded some plants from the sorghum 2022 dataset because of interactions between biology and phenotyping system, which resulted in images that were too challenging to annotate. Specifically, these plants produced long and non-erect leaves where the tips lay on the floor of the imaging chamber, which caused the following problems: (1) some leaf tips would be hidden inside the turntable mechanism, (2) some leaf tips would move too much between shots because they were rubbing the floor, and (3) eventually after rubbing the floor for too long, some leaf tips would break or split.

The hyper-parameter θ is critical to getting good results. As shown in Section 3.1, the best value for θ varies according to the camera setup, the amount of noise, and occlusion. It is difficult to tune its value for a certain dataset because it is complex to measure the amount of noise and occlusion accurately. Instead, we recommend estimating the best value for θ with a training set for which the ground truth is known. In general, increasing the value of θ will favor matching between points and will tend to decrease the number of leaves discovered. Conversely, decreasing the value of θ will prevent matching between points and increase the number of leaves discovered. The best value for θ depends on the amount of noise of point re-projections: more noise means that the similarity between two similar points will decrease. Therefore θ needs to be increased. In this case, the best value for θ is the lowest value such that matching is done correctly.

5. Conclusion

We presented an approximate solution to the joint multi-view sparse correspondence and triangulation problem. Our dynamic programming algorithm finds a local minimum of the total re-projection error on all views by recursively solving overlapping sub-problems and merging their results into a final solution. We show that our approach is robust to noise and occlusion with experiments on simulated data and real datasets. In particular, we apply our algorithm to phenotyping and show that it can aggregate the information from multiple views to improve the accuracy of leaf counting with sorghum.

The complexity of our algorithm with respect to the number of 3D points is polynomial. However, the number of views is critical, and it exponentially impacts the algorithm's speed. Nevertheless, our algorithm is suitable for high-throughput phenotyping because plants have a limited number of phenotypes, and imaging setups take a limited number of images.

The optimal number of views is a tradeoff between accuracy and running time. Adding views improve triangulation accuracy and mitigate the effects of noise and occlusion. But more views take exponentially more time to process, and the gain in accuracy plateaus out.

We demonstrate our implementation on an imaging setup with two cameras and a turntable, taking 6–10 asynchronous views of a plant, assuming a large gap between views. Our solution may not be suitable for problems with more than 15 views, *e.g.,* a video sequence shot with one camera, because it does not use frame coherence between views.

In future work, it would be possible to improve both the speed and accuracy of the multi-view triangulation function, which is the biggest bottleneck. Our implementation is single core, but it can be parallelized by running it bottom-up: first, all subsets of two views in parallel, then all subsets of three views in parallel, etc.

An interesting avenue for future work would be to change the phenotypes we triangulate to curves or line segments rather than only points. This way, the matching would be more robust, allowing the reconstruction of 3D skeletons from 2D views.

CRediT authorship contribution statement

Mathieu Gaillard: Conceptualization, Methodology, Software, Validation, Formal analysis, Data curation, Writing – original draft. **Bedrich Benes:** Writing – review & editing, Supervision, Funding acquisition. **Michael C. Tross:** Investigation, Writing – review & editing. **James C. Schnable:** Writing – review & editing, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Bedrich Benes reports financial support was provided by Foundation for Food and Agriculture Research.

Data availability

Data will be made available on request.

Acknowledgments

This research was supported by the Foundation for Food and Agriculture Research, United States Grant ID: 602757 to Benes and Schnable. The content of this publication is solely the responsibility of the authors and does not necessarily represent the official views of the foundation for Food and Agriculture Research. This project was completed utilizing the Rosen Center for Advanced Computing (RCAC) (McCartney et al., 2014) of Purdue University. This publication uses data generated via the Computer Vision Annotation Tool (CVAT) platform, development of which is funded by Intel. We thank Leighton Wheeler, Nate Pester, Isaac Stevens, Elijah Frost, Thomas Hoban, Sierra Conway, Elliot Kadrofske, Emma Chrzanowski, Carolina Freitas, Lou Townsend, and Logan Duryee for annotating leaf tips in the sorghum 2018 dataset. We thank Vincent Stoerger and Troy Pabst for growing and imaging plants of the sorghum 2022 dataset, as well as Ryleigh Grove and Cole Hammett for counting the leaves on the live plants during summer 2022. Finally, we thank Chenyong Miao for insightful discussions and for motivating this work.

Appendix A. Full pipeline evaluation

The aim of this appendix is to demonstrate the accuracy of the full reconstruction pipeline using cheap equipment. A LEGO® object was chosen because it is precisely engineered and readily available. Although in this example, the correspondence problem is not particularly challenging, the accuracy of the final measurement depends on a few factors, including image acquisition, camera calibration, and multi-view triangulation, all of which can introduce numerical errors.

Five pictures (Fig. A.12) of a LEGO® set were taken with a Google Pixel 3 phone. All pictures were calibrated using a ChArUco pattern (Garrido-Jurado et al., 2014). We manually annotated the two extremities of a 2 × 4 LEGO® plate (3020) in each image.

Our algorithm took as input the five images, the five sets of 2D points, and output the estimated length of the plate. Our result is 31.7 mm, which is a 0.3% error from the actual length of 31.8 mm measured by Bartneck (2019). This measurement pipeline with an object on a calibration pattern could be replicated to occasionally measure a plant.

Appendix B. Additional figures

This appendix include figures in addition to the main paper to give additional details about experiments.

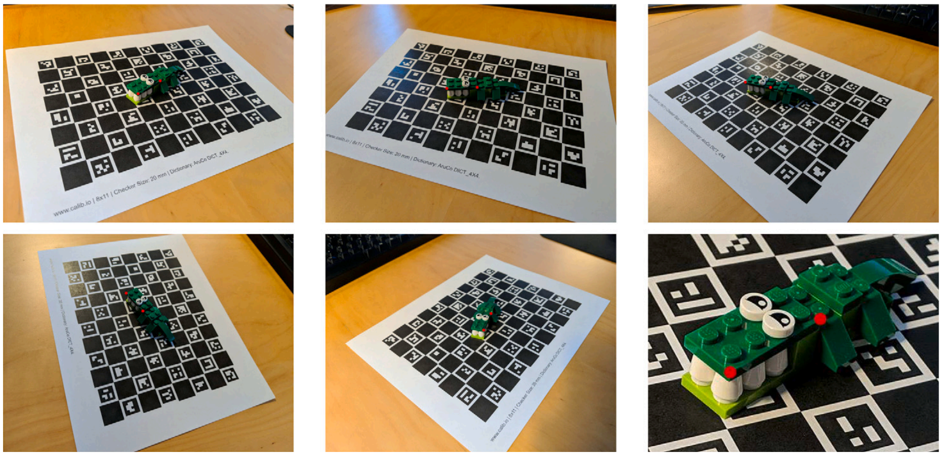
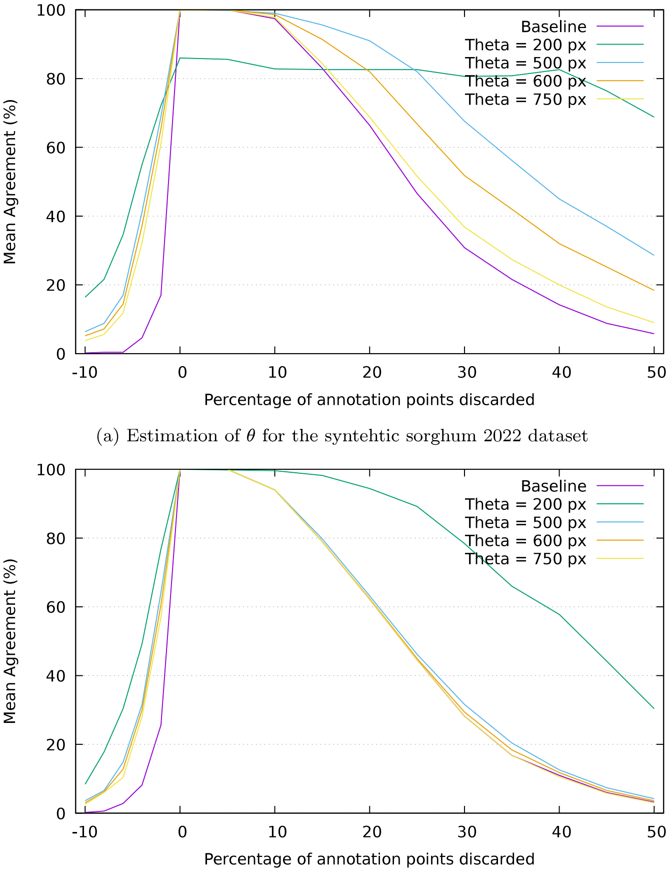


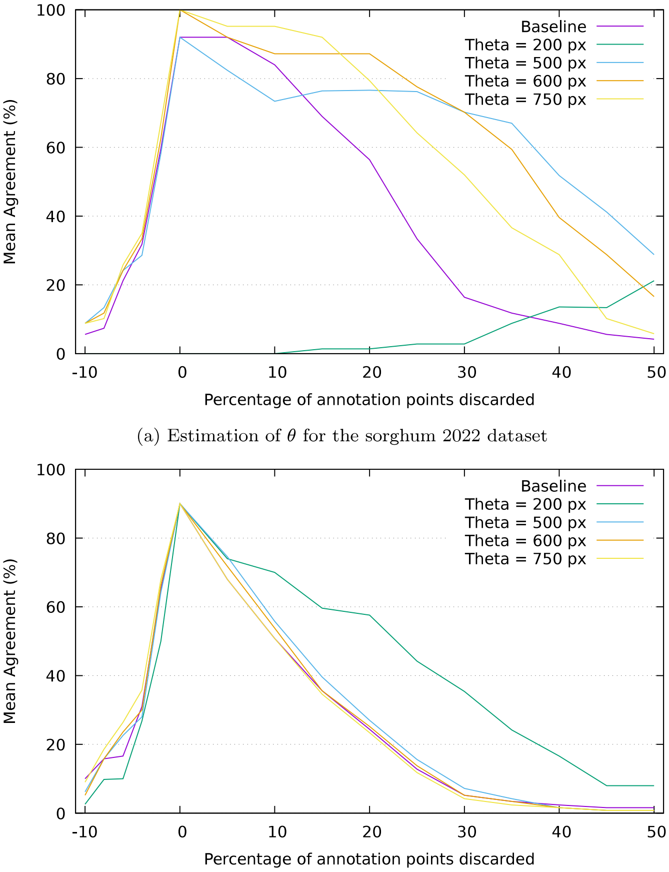
Fig. A.12. The LEGO® crocodile, whose snout is composed of a 2 × 4 plate (3020). The figure shows five views and a zoom in of the first view. Red points show the manual annotation of the two extremities of the plate. Our algorithm successfully corresponded points and was able to precisely measure the length of the plate with only 0.3% error (31.7 mm vs 31.8 mm the ground truth).



(a) Estimation of θ for the syntehtic sorghum 2022 dataset

(b) Estimation of θ for the syntehtic sorghum 2018 dataset

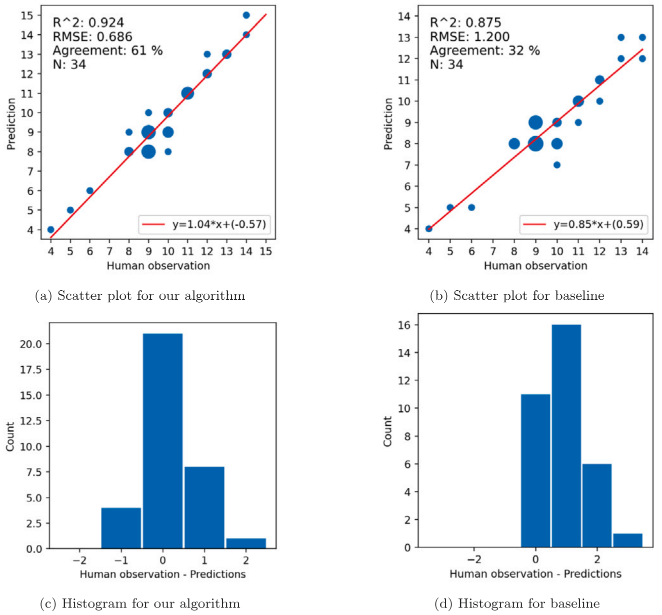
Fig. A.13. Estimation of the theoretical best value for the hyper-parameter θ for both imaging setups. The plot shows the same experiment as in (a) of Fig. A.14 but with an ideal synthetic dataset. The dataset consists of 100 plants with 5 to 15 leaves imaged with both imaging setups, without noise and occlusion. This dataset represents the best case scenario for both imaging setups and gives an upper bound on what can be achieved by our algorithm. With the 2020 imaging setup, on the contrary to the real dataset, the best value for the threshold is $\theta = 500$ px (instead of $\theta = 750$ px). This can be explained by the fact that the synthetic dataset has perfect camera calibration and no noise. Note that the value $\theta = 200$ px significantly outperforms the baseline when the percentage of changed annotations is not in $[0\%, 20\%]$.



(a) Estimation of θ for the sorghum 2022 dataset

(b) Estimation of θ for the sorghum 2018 dataset

Fig. A.14. Estimation of the best value for the hyper-parameter θ for both the 2018 and 2022 datasets. The same experiment as in Fig. 11 is run for different values of θ. Curves show the mean agreement w.r.t. the percentage of changed annotations for $\theta \in \{200, 500, 600, 750\}$. Higher values of θ tend to give results that are closer to the baseline, and lower values of θ tend to give better results when lots of annotations are discarded, but worse results otherwise. The best value leads to a mean agreement that is always better than the baseline. When annotations are added, our algorithm only performs marginally better than the baseline. It is however never significantly worse than the baseline. For the 2022 dataset, the best value is $\theta = 750$ px ($\theta = 600$ px could also work). For the 2018 dataset, the best value is $\theta = 200$ px.



(a) Scatter plot for our algorithm

(b) Scatter plot for baseline

(c) Histogram for our algorithm

(d) Histogram for baseline

Fig. A.15. Relationship between predicted leaf number and human annotated leaf number for our algorithm and the baseline on both 2018 and 2022 datasets with 20% of annotations discarded. For reference, this figure shows the same plots as in Fig. 10, except that 20% of annotations are discarded. Our algorithm outperforms the baseline with an agreement of 61% versus 32%. Also, the histograms of the distribution of errors show that even when the leaf number is not predicted correctly, the value is still close to the ground truth. This is reflected in the RMSE, that is 0.686 with our algorithm as opposed to 1.2 for the baseline. In other words, when our algorithm makes an error, it is a smaller error than with the baseline approach.

References

Anwar, S., Khan, S., Barnes, N., 2020. A deep journey into super-resolution: A survey. ACM Comput. Surv. 53 (3), http://dx.doi.org/10.1145/3390462.

Bartneck, C., 2019. LEGO brick dimensions and measurements. https://www.bartneck.de/2019/04/21/lego-brick-dimensions-and-measurements/, (Accessed: 2021-11-16).

Bedekar, A.S., Haralick, R.M., 1996. Finding corresponding points based on Bayesian triangulation. In: Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition. pp. 61–66. http://dx.doi.org/10.1109/CVPR.1996.517054.

Bernard, F., Thunberg, J., Goncalves, J., Theobalt, C., 2019. Synchronisation of partial multi-matchings via non-negative factorisations. Pattern Recognit. 92, 146–155. http://dx.doi.org/10.1016/j.patcog.2019.03.021.

Boogaard, F.P., Rongen, K.S., Kootstra, G.W., 2020. Robust node detection and tracking in fruit-vegetable crops using deep learning and multi-view imaging. Biosyst. Eng. 192, 117–132. http://dx.doi.org/10.1016/j.biosystemseng.2020.01.023.

Bradski, G., 2000. The OpenCV library. Dr. Dobb's J. Softw. Tools.

Brooking, I.R., Jamieson, P.D., Porter, J.R., 1995. The influence of daylength on final leaf number in spring wheat. Field Crops Res. 41 (3), 155–165. http://dx.doi.org/10.1016/0378-4290(95)00014-H.

Chen, Y., Guibas, L., Huang, Q., 2014. Near-optimal joint object matching via convex relaxation. In: Intl. Conference on Machine Learning. pp. 100–108. http://dx.doi.org/10.48550/arXiv.1402.1473.

Chen, J., Wu, D., Song, P., Deng, F., He, Y., Pang, S., 2020. Multi-view triangulation: Systematic comparison and an improved method. IEEE Access 8, 21017–21027. http://dx.doi.org/10.1109/ACCESS.2020.2969082.

Cheng, Y.-Q., Collins, R.T., Hanson, A.R., Riseman, E.M., 1994. Triangulation without correspondences. In: Proceedings of ARPA Image Understanding Workshop. pp. 993–1000.

Dellaert, F., 2001. Monte carlo em for data-association and its applications in computer vision (Ph.D. thesis). Carnegie Mellon.

Edmonds, J., Karp, R.M., 1972. Theoretical improvements in algorithmic efficiency for network flow problems. J. ACM 19 (2), 248–264.

Fahlgren, N., Feldman, M., Gehan, M.A., Wilson, M.S., Shyu, C., Bryant, D.W., Hill, S.T., McEntee, C.J., Warnasooriya, S.N., Kumar, I., et al., 2015. A versatile phenotyping system and analytics platform reveals diverse temporal responses to water availability in Setaria. Molecular Plant 8 (10), 1520–1535.

Fathian, K., Khosoussi, K., Tian, Y., Lusk, P., How, J.P., 2020. CLEAR: A consistent lifting, embedding, and alignment rectification algorithm for multiview data association. IEEE Trans. Robot. 36 (6), 1686–1703. http://dx.doi.org/10.1109/TRO.2020.3002432.

Fischler, M.A., Bolles, R.C., 1981. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM 24 (6), 381–395. http://dx.doi.org/10.1145/358669.358692.

Gaillard, M., Miao, C., Schnable, J.C., Benes, B., 2020. Voxel carving-based 3D reconstruction of sorghum identifies genetic determinants of light interception efficiency. Plant Direct 4 (10), e00255. http://dx.doi.org/10.1002/pld3.255.

Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F.J., Marín-Jiménez, M.J., 2014. Automatic generation and detection of highly reliable fiducial markers under occlusion. Pattern Recognit. 47 (6), 2280–2292. http://dx.doi.org/10.1016/j.patcog.2014.01.005.

Ge, Y., Bai, G., Stoerger, V., Schnable, J.C., 2016. Temporal dynamics of maize plant growth, water use, and leaf water content using automated high throughput RGB and hyperspectral imaging. Comput. Electron. Agric. 127, 625–632.

Hammer, G.L., van Oosterom, E., McLean, G., Chapman, S.C., Broad, I., Harland, P., Muchow, R.C., 2010. Adapting APSIM to model the physiology and genetics of complex adaptive traits in field crops. J. Exp. Bot. 61 (8), 2185–2202. http://dx.doi.org/10.1093/jxb/erq095.

Han, X., Laga, H., Bennamoun, M., 2019. Image-based 3d object reconstruction: State-of-the-art and trends in the deep learning era. IEEE Trans. Pattern Anal. Mach. Intell. http://dx.doi.org/10.1109/TPAMI.2019.2954885.

Hartley, R.I., Sturm, P., 1997. Triangulation. Comput. Vis. Image Underst. 68 (2), 146–157. http://dx.doi.org/10.1006/cviu.1997.0547.

Hartley, R., Zisserman, A., 2003. Multiple View Geometry in Computer Vision. Cambridge univ. Press, http://dx.doi.org/10.1017/CBO9780511811685.

He, J., Le Gouis, J., Stratonovitch, P., Allard, V., Gaju, O., Heumez, E., Orford, S., Griffiths, S., Snape, J.W., Foulkes, M.J., Semenov, M.A., Martre, P., 2012. Simulation of environmental and genotypic variations of final leaf number and anthesis date for wheat. Eur. J. Agron. 42, 22–33. http://dx.doi.org/10.1016/j.eja.2011.11.002, Designing Crops for new challenges.

Huang, Q.-X., Guibas, L., 2013. Consistent shape maps via semidefinite programming. In: Symposium on Geometry Processing. SGP '13, Eurographics Association, pp. 177–186.

Junker, A., Muraya, M.M., Weigelt-Fischer, K., Arana-Ceballos, F., Klukas, C., Melchinger, A.E., Meyer, R.C., Riewe, D., Altmann, T., 2015. Optimizing experimental procedures for quantitative evaluation of crop plant performance in high throughput phenotyping systems. Front. Plant Sci. 5, 770.

Kuhn, H.W., 1955. The Hungarian method for the assignment problem. Nav. Res. Logist. Q. 2 (1–2), 83–97.

Lizaso, J., Batchelor, W., Westgate, M., 2003. A leaf area model to simulate cultivar-specific expansion and senescence of maize leaves. Field Crops Res. 80 (1), 1–17. http://dx.doi.org/10.1016/S0378-4290(02)00151-X.

Lowe, D.G., 2004. Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. 60 (2), 91–110. http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94.

Lv, H., Chen, Z., Mo, Y., Lou, L., Song, R., Doonan, J.H., 2022. Segmentation and counting of plant organs using deep learning and multi-view images. In: Jansen, T., Jensen, R., Mac Parthaláin, N., Lin, C.-M. (Eds.), Advances in Computational Intelligence Systems. Springer International Publishing, Cham, pp. 406–411. http://dx.doi.org/10.1007/978-3-030-87094-2_36.

Maciel, J., Costeira, J.P., 2003. A global solution to sparse correspondence problems. IEEE Trans. Pattern Anal. Mach. Intell. 25 (2), 187–199. http://dx.doi.org/10.1109/TPAMI.2003.1177151.

Maset, E., Arrigoni, F., Fusiello, A., 2017. Practical and efficient multi-view matching. In: Proceedings of the IEEE International Conference on Computer Vision. ICCV, pp. 4568–4576. http://dx.doi.org/10.1109/ICCV.2017.489.

McCartney, G., Hacker, T., Yang, B., 2014. Empowering faculty: A campus cyberinfrastructure strategy for research communities. Educause Rev. URL https://er.educause.edu/articles/2014/7/empowering-faculty-a-campus-cyberinfrastructure-strategy-for-research-communities.

Miao, C., Guo, A., Thompson, A.M., Yang, J., Ge, Y., Schnable, J.C., 2021. Automation of leaf counting in maize and sorghum using deep learning. Plant Phenome J. 4 (1), e20022.

Okutomi, M., Kanade, T., 1993. A multiple-baseline stereo. IEEE Trans. Pattern Anal. Mach. Intell. 15 (4), 353–363. http://dx.doi.org/10.1109/34.206955.

Pachauri, D., Kondor, R., Singh, V., 2013. Solving the multi-way matching problem by permutation synchronization. In: Advances in Neural Information Processing Systems. Curran Associates Inc., Red Hook, NY, USA, pp. 1860–1868, URL https://proceedings.neurips.cc/paper/2013/file/3df1d4b96d8976ff5986393e8767f5b2-Paper.pdf.

Pound, M.P., Atkinson, J.A., Townsend, A.J., Wilson, M.H., Griffiths, M., Jackson, A.S., Bulat, A., Tzimiropoulos, G., Wells, D.M., Murchie, E.H., Pridmore, T.P., French, A.P., 2017. Deep machine learning provides state-of-the-art performance in image-based plant phenotyping. GigaScience 6 (10), http://dx.doi.org/10.1093/gigascience/gix083, gix083.

Roy, S., Cox, I.J., 1998. A maximum-flow formulation of the n-camera stereo correspondence problem. In: Sixth Intl. Conference on Computer Vision (IEEE Cat. No. 98CH36271). IEEE, pp. 492–499. http://dx.doi.org/10.1109/ICCV.1998.710763.

Scott, G.L., Longuet-Higgins, H.C., 1991. An algorithm for associating the features of two images. Proc. R. Soc. B 244 (1309), 21–26. http://dx.doi.org/10.1098/rspb.1991.0045.

Shapiro, L.S., Brady, J.M., 1992. Feature-based correspondence: an eigenvector approach. Image Vis. Comput. 10 (5), 283–288. http://dx.doi.org/10.1016/0262-8856(92)90043-3.

Shi, W., van de Zedde, R., Jiang, H., Kootstra, G., 2019. Plant-part segmentation using deep learning and multi-view vision. Biosyst. Eng. 187, 81–95. http://dx.doi.org/10.1016/j.biosystemseng.2019.08.014.

Tollenaar, M., Hunter, R.B., 1983. A photoperiod and temperature sensitive period for leaf number of maize. Crop Sci. 23 (3), http://dx.doi.org/10.2135/cropsci1983.0011183X002300030004x, cropsci1983.0011183X002300030004x.

Triggs, B., McLauchlan, P.F., Hartley, R.I., Fitzgibbon, A.W., 1999. Bundle adjustment—a modern synthesis. In: Intl. Workshop on Vision Algorithms. Springer, pp. 298–372. http://dx.doi.org/10.1007/3-540-44480-7_21.

Truong, S.K., McCormick, R.F., Mullet, J.E., 2017. Bioenergy sorghum crop model predicts VPD-limited transpiration traits enhance biomass yield in water-limited environments. Front. Plant Sci. 8, http://dx.doi.org/10.3389/fpls.2017.00335.

Xiao, D., Li, J., Li, K., 2019. Robust precise dynamic point reconstruction from multi-view. IEEE Access 7, 22408–22420. http://dx.doi.org/10.1109/ACCESS.2019.2896096.

Yilmaz, A., Javed, O., Shah, M., 2006. Object tracking: A survey. Acm Comput. Surv. (CSUR) 38 (4), 13–es. http://dx.doi.org/10.1145/1177352.1177355.

Yu, J.-G., Xia, G.-S., Samal, A., Tian, J., 2016. Globally consistent correspondence of multiple feature sets using proximal Gauss–Seidel relaxation. Pattern Recognit. 51, 255–267. http://dx.doi.org/10.1016/j.patcog.2015.09.029.

Zhou, X., Zhu, M., Daniilidis, K., 2015. Multi-image matching via fast alternating minimization. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 4032–4040. http://dx.doi.org/10.1109/ICCV.2015.459.