Plant Leaf Meshes from Time-of-Flight RGB-D Sensors

Anonymous 3DV submission

Paper ID ****

Abstract

1. Introduction

This paper addresses the problem of automatically building 3D mesh models of plant leafs using inexpensive time-of-flight RGB-D sensors. Plant researchers, seeking to understand genetic underpinnings of plant growth [REF] and seeking to develop new varieties [REF], need automated ways to non-invasively measure plant phenotypes including growth, leaf distributions, orientations, photosynthesis and productivity [REF]. An important step in estimating all of these properties is obtaining 3D shape and pose for all the plant leafs. Plants cannot be moved or disturbed in growth chambers, and so our concept is to mount close-range RGB-D sensors in the chambers and acquire 3D mesh models of the leafs.

Time-of-flight RGB-D sensors have both advantages and drawbacks compared to other 3D sensors. [Expand the below]

- Dense depth image without scanning or rotating the target
- Depth even on non-textured regions
- Closely-aligned high resolution color image
- Near IR reflectance image
- Significantly higher depth error than laser scanners
- Short range and sensitive to specular artifacts
- Occlusion boundary artifacts (as with most ranging devices)

Here we address a key obstacle in obtaining accurate leaf models: the large noise in the depth images. [Expand the below:]

- There has been research in merging depth maps from multiple views and in reducing noise this way. This is not so helpful here as the sensor is fixed, and the size of the noise compared to the features we want to observe is large in our application.
- More ...

Figure 1. Time-of-flight RGB-D sensor consists of a flash IR emitter, an IR sensor that measures time-of-travel for the reflected light on a dense pixel grid, and a color camera. In this paper the Creative Senz3D sensor was used.

Our solution is a new mesh generation algorithm that leverages the high resolution color image, the dense depth estimates and the near-IR reflection image to overcome large depth errors. The paper is organized as follows. [Explain]

2. Related Work

3. Sensor Data Characterization

We used a Creative Senz3D RGB-D sensor [1]. The sensor contains both a 1280×720 color camera adjacent to a depth camera with a resolution of 320×240 pixels. A flash IR emitter illuminates the scene and the adjacent IR sensor measures the time-of-travel for the reflected light at a dense 320×240 pixel grid, producing a dense depth map. This sensor operates in a similar way to the Kinect 2 but is designed for closer range targets.

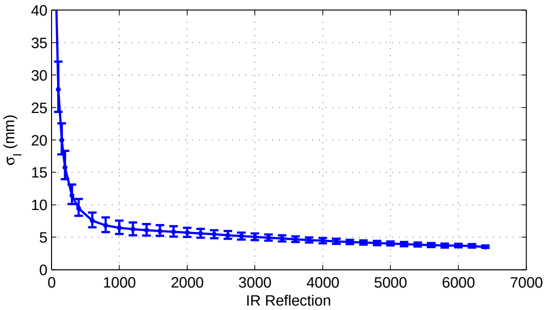


Figure 2. Image-varying noise is predicted well by the IR reflectance in raw units returned by the camera.

While the sensor produces dense depth measurements over target leaf surfaces, the difficulty in converting these measurements into 3D models is that the noise in the measurements is significantly larger than the features we are seeking to recover. In this section we model and quantify the measurement noise.

3.1. Noise Characterization

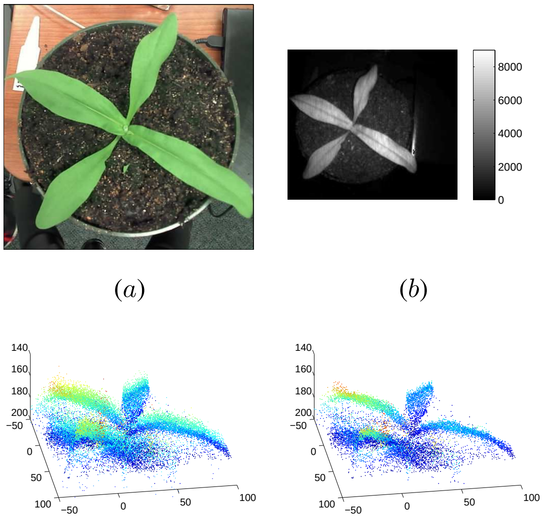
The depth camera returns an IR reflectance in addition to a depth value at each pixel. Hence it can be calibrated in the same way as the color camera using Zhang's method [2]. The result is that each pixel defines a ray from the camera. Depth noise is modeled as a one dimensional random variable, ε, for each pixel along its ray direction.

The depth noise, ε, is modeled as the sum of an image-varying term, ε_I, and a scene-varying term, ε_S:

$$\varepsilon = \varepsilon_I + \varepsilon_S. \tag{1}$$

The term ε_I models the random change in depth for camera pixels of subsequent images of a static scene from a static camera. To quantify this term we measured the standard deviation σ_I in depth of each pixel for a batch of 300 images of a fixed scene containing a flat matte surface. We repeated this at different poses and depths, and with different surface albedos. While target depth, inclination, albedo, and pixel position are all correlated with σ_I, we found that the best predictor for σ_I was the IR reflectance intensity, as shown in Figure 2. For typical scenes the single measurement noise in depth is roughly 5mm, although for low reflectivity objects or objects at long range this noise can increase significantly. Fortunately plant leafs are good IR reflectors.

Averaging depth measurements of a fixed scene will reduce the noise from ε_I, but will not reduce the noise from ε_S. This latter scene-varying term is constant for a static scene, but changes when the scene changes. To characterize this noise we first eliminated (approximately) the image-varying noise contribution by averaging over a large number of images (300). Then assuming ε_S is independent and identically distributed between pixels, we measured the variance of the pixel depth errors between a known flat surface and the estimated surface. In our experiments we obtained $\sigma_S = 6.5mm$, and found that it was insensitive to changes in depth.



(a) (b)

(c) (d)

Figure 3. Illustration of sensor data. (a) Portion of color image. (b) IR reflectance image. (c) Portion of a single depth image surrounding plant projected into 3D showing significant depth noise. (d) Average of 60 depth images projected into 3D, with σ_S being the dominant source of noise. Units of 3D plots are mm.

The total pixel noise can be estimated assuming independence of ε_I and ε_S, and is given by:

$$\sigma^2 = \frac{\sigma_I^2}{N} + \sigma_S^2, \tag{2}$$

where N is the number of images averaged over. When averaging 5 or more depth images the scene-varying contribution, σ_S^2, will dominate. There are additional sources of noise not modeled by this. These include object specularities, and mixed-depth pixels on object edges. These tend to produce very large image-varying noise, σ_I, and we discard these points.

4. Mesh Fitting

We pose mesh fitting to 3D point data as finding the most likely surface that would have generated those points. By incorporating prior surface assumptions, the fitting process estimates a continuous surface from discrete points that can eliminate much of the measurement noise. Methods that fit mesh models to 3D points often minimize the perpendicular distance of points to facets [REF]. This makes sense when point-cloud noise is equal in all directions or else the point noise is small compared to the facets. For our data the measurement noise is large and is not equal in all directions, but rather is along the depth camera rays. Hence the focus of this section is to develop a mesh fitting method that minimizes these pixel depth errors along the pixel rays.

In this paper we define a mesh in a 2D image space and project it into 3D. This is more limiting than full 3D meshes as it models only the surface portions visible from the sensor, but it also provides a number of advantages. Compared to methods that fit prior surface models to depth maps [ref], need to search of the space of poses, scales and distortions of the model with the chance of finding local minima. Compared to voxel-based models with implicit surfaces [ref], our method can better incorporate pixels uncertainties and surface priors, as well as having fewer discretization artifacts. In addition our method can naturally incorporate detailed features from the high-resolution color camera, and reflectivity information from the IR reflectance image.

4.1. Notation

A point, $\boldsymbol{p}$, is a vector in 3D. In a given camera coordinate system, it projects onto a pixel on the unit focal-length image-plane $\hat{\boldsymbol{p}} = (u, v, 1)^\top$, where the " $\hat{}$ " indicates a homogeneous vector, and u and v are the coordinates in this plane. Now $\hat{\boldsymbol{p}}$ defines a ray from the camera origin, and the original point is obtained by scaling the image-plane point by its depth, λ, along the ray, namely: $\boldsymbol{p} = \lambda \hat{\boldsymbol{p}}$.

4.2. Facet Model

Mesh fitting for an individual facet is illustrated in Figure 4. The 2D vertices and edge connections are determined in an image, in this case the color image although it could be the depth image, as described in section 5. If these vertices lie on a feature of the target leaf, such as its edge, we know that that those 3D features like somewhere along the rays emanating from camera origin through those vertices. Hence a triangular facet approximation to the object surface will have vertices on these three rays.

The next step is to associate depth measurements with the facet. Pixels in the depth camera are projected along their rays out into 3D. The resulting 3D points that lie within the facet pyramid (defined by these rays through its vertices) will project into the 2D facet in the color image. Hence it is straightforward to associate 3D points with mesh facets.

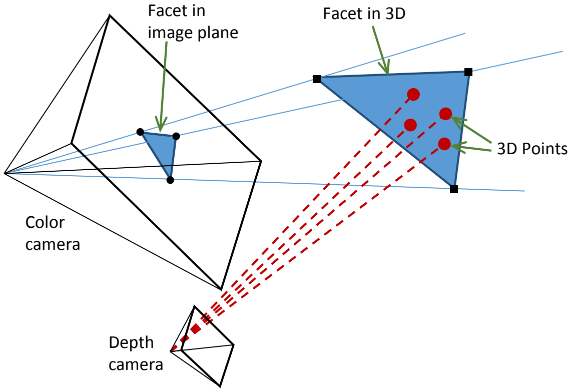


Figure 4. The parallel and adjacent color and depth cameras are shown as pyramids denoting their fields of view, and their size difference illustrates their relative resolutions. Three vertices in a color image define the rays on which the vertices of the corresponding 3D object facet must lie. This facet is fit using the the 3D points projected out from the depth camera.

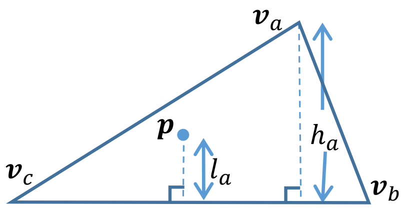


Figure 5. The coordinates of a point on a facet described by Eq. (3) are the weighted linear sum of the three vertex coordinates. The weight α_i for vertex $\boldsymbol{v}_i$ is the ratio of its perpendicular distance l_i to the opposite edge to the vertex perpendicular distance h_i. Analogous expressions describe α_j and α_k.

To estimate the facet parameters from depth measurements we will express the depth points as a linear function of the vertices of its facet. We make a local orthographic approximation for the projection of a facet. This will be a good approximation as long as the facet size is small compared to is depth from the camera, which is true for most applications. Given this assumption, the coordinates of a point $\boldsymbol{p}$ lying on a facet can be expressed as a linear combination of the three vertex coordinates $\boldsymbol{v}_i$, $\boldsymbol{v}_j$ and $\boldsymbol{v}_k$ as follows:

$$\boldsymbol{p} = \alpha_i \boldsymbol{v}_i + \alpha_j \boldsymbol{v}_j + \alpha_k \boldsymbol{v}_k. \tag{3}$$

The coefficients α_i, α_j and α_k are defined in Figure 5. This equation applies both to the image coordinates of the point and vertices, and the 3D point and 3D vertices. The coefficients can be computed from this equation in image coordinates as the point and vertices image coordinates are known. In 3D we can select the third component of this equation and write it:

$$\lambda_p = \alpha_i \lambda_i + \alpha_j \lambda_j + \alpha_k \lambda_k, \tag{4}$$

where λ_i is the third component of $\boldsymbol{v}_i$ and so forth.

4.3. Least Squares

Equation (4) gives the depth of one point in terms of its facet vertices. For mesh with many facets and a measurement with many depth points, a vector of pixel depths, $\boldsymbol{\lambda}_d$, and vector of vertex depths, $\boldsymbol{\lambda}_v$, are related with a coefficient matrix, A, containing the appropriate α's:

$$\boldsymbol{\lambda}_d = A\boldsymbol{\lambda}_v. \tag{5}$$

Given a measurement vector of point depths, $\tilde{\boldsymbol{\lambda}}_d$, expressed in the color camera coordinates, an error vector between these depths and the corresponding mesh points is: $\tilde{\boldsymbol{\lambda}}_d - A\boldsymbol{\lambda}_v$. Notice that this error is along the color camera rays which are almost parallel to the depth camera rays, and thus to a good approximation the noise model in Eq. (2) applies. This leads to the following weighted squared error formulation:

$$E_{depth} = \|W\tilde{\boldsymbol{\lambda}}_d - WA\boldsymbol{\lambda}_v\|^2, \tag{6}$$

where W is a diagonal matrix containing the inverse standard deviation, σ^{-1}, from Eq. (2).

4.4. Regularization

Prior models on surface properties can be incorporated into the mesh via regularization and in so doing reduce the impact of noise. Membrane energy is a well-used function in mesh optimization [?] and can be minimized using the discrete Laplacian operator. Here we use Laplacian smoothing due to its simplicity and good performance [?, ?, ?], although we modify it to accommodate image-based edge information. In addition, rather than apply Laplacian smoothing after the fact, entailing an iterative optimization [?], we show that Laplacian smoothing can be incorporated directly into the least squares mesh estimation. This has a number of advantages over application after the initial mesh estimation. First the smoothing penalty is traded off against measurement error rather than vertex offset. Second, the due to our ray constraints on the vertices we are able to derive a linear solution with not need to iterate. Finally when the regularization components are added to Eq. (6) they ensure that the solution is well-posed even when some of the facets have no depth points in them.

Laplacian smoothing uses an umbrella-operator [?] on a vertex, $\boldsymbol{v}$, and its neighbors $\boldsymbol{v}_i \in \mathcal{N}(\boldsymbol{v})$,

$$\boldsymbol{u}(\boldsymbol{v}) = \frac{1}{n}\sum_{\boldsymbol{v}_i \in \mathcal{N}} \boldsymbol{v}_i - \boldsymbol{v}, \tag{7}$$

for n neighbors as illustrated in Figure 6. In our model the vertices lie along known rays and so this operator can be expressed as a function of the vertex depth: $\boldsymbol{u}(\lambda) = \frac{1}{n}\sum_{i\in\mathcal{N}} \lambda_i \hat{\boldsymbol{v}}_i - \lambda\hat{\boldsymbol{v}}$. The squared magnitude $\|\boldsymbol{u}(\lambda)\|^2$ is a natural penalty term as it captures a discrete form of the membrane energy. Summing this over all vertices and arranging the known $\hat{\boldsymbol{v}}$ components into a single matrix U, we obtain

$$E_{reg} = \|U\boldsymbol{\lambda}\|^2. \tag{8}$$

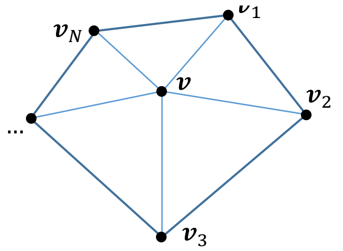


Figure 6. In discrete form the Laplacing is implemented as an umbrella operator, Eq. (7), over a vertex $\boldsymbol{v}$ and its first neighbors.

5. Mesh Initialization

6. Results

7. Conclusion

References

[1] V. Nguyen, M. Chew, and S. Demidenko. Vietnamese sign language reader using intel creative senz3d. In *IEEE International Conference on Automation, Robotics and Applications (ICARA)*, pages 77–82, 2015. 1

[2] Z. Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, Nov 2000. 2