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 leaves 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 limited-space growth chambers making it impractical to use scanning lasers for shape modeling. Instead our concept is to mount close-range time-of-flight RGB-D sensors in the chambers, acquire color, IR-reflectance and range data, and estimate leaf shape through building 3D mesh models of the leafs.

While time-of-flight RGB-D sensors are small, inexpensive and provide dense 3D surface sampling of objects, they pose a challenge to surface modeling of small objects due to large depth noise. This pixel-depth noise is significantly larger than the surface features we hope to capture, and can be on the order of the leaf size. The goal of this paper is to cut through this noise to obtain high fidelity surface models of leaves and small objects.

Mesh fitting to 3D point clouds and depth maps has had a number of objectives. A mesh provides a surface topology and geometry to a point cloud [11, 13]. This enables visualization and shape analysis of the target object. A mesh can efficiently compress dense scanned data and mesh fitting methods have been developed that adhere to the underlying shape and preserve geometric features such as creases [4, 6]. The problem differs from these in that our depth data are far noisier and our goal is to recover the true surface rather than adhering as closely as possible to the 3D points.

Another goal for mesh fitting is to incrementally build full 3D models of objects. Zippered polygons [12] builds

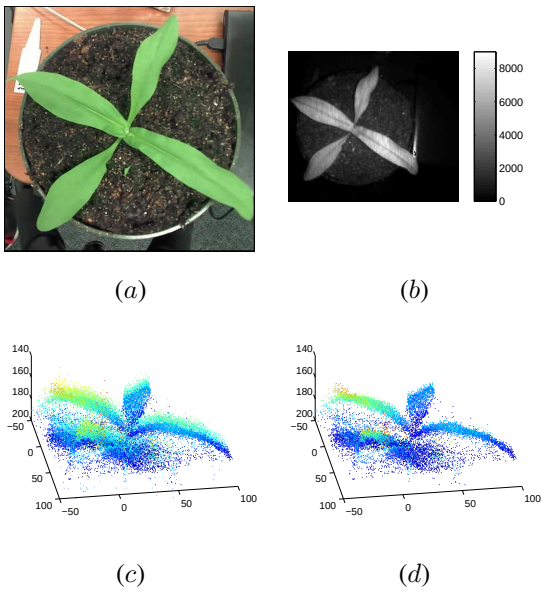

Figure 1. Illustration of sensor data. (a) Portion of color image. (b) IR reflectance image with reflectance values. (c) Portion of a single depth image surrounding plant projected into 3D showing significant depth noise. (d) Average of 60 depth images projected into 3D, with σ_S being the dominant source of noise. Units of 3D plots are mm. The goal of this paper is to combine these data elements into

mesh models from range images and combines them discarding noisy points at the boundaries. The method developed by Curless and Levoy [3] populates a weighted voxel occupancy grid from the depth data and recovers the surface by triangulating an isosurface. By raycasting this surface, new camera poses can be aligned and their depths maps contribute to and improve the voxel model. Advantages of this method include that surface topology is automatically determined, additional data can be readily incorporated, holes can be filled when more data are collected and it incorporates directional uncertainty of range data into the models. Recent approaches for environment modeling from RGB-D cameras such as Kinect Fusion [5, 8] build on this voxel modeling and accumulation. For our application the sensor

is fixed and there is no option for merging views from different perspectives. Artifacts caused by discretization into voxels are significant particularly with high input noise, and the method is limited in incorporating surface smoothness priors.

The method proposed here is a new mesh generation algorithm that resolves object shape despite large noise in the range images by leveraging multiple sensor modalities. Edge features of the high resolution color image are used to help select and constrain vertices and mesh edges. The errors in depth are modeled along camera rays and minimized during fitting to facets. The IR reflectance image is used as a Shape from Shading cue to influence facet surface normals. A linear least squares solution to Laplacian smoothing is integrated into the mesh fitting for regularization with surface curvature priors, and modification is made to preserve sharp edges and creases.

2. Related Work

Optimizing meshes for fit point cloud data has been approached through vertex additions and removals [4], although this work assumes the point data are precise and dense

3. Sensor Data Characterization

We explore using a new class of RGB-D sensor such as the Creative Senz3D [9] to build 3D mesh models of plant leafs. The sensor contains a high resolution color camera (1280×720 pixels) adjacent and parallel to a lower resolution depth camera (320×240 pixels). A flash IR emitter adjacent to these cameras illuminates the scene and depth sensor measures the time-of-travel for the reflected light as well as its reflectance over its pixel grid. These modalities are illustrated in Figure 1. It operates in a similar way to the Kinect 2 but is designed for closer range targets.

While the sensor produces dense depth measurements over target leaf surfaces, the noise in depth measurements is significantly larger than the features we are seeking to recover as illustrated in Figure 1(c). A key goal in this paper is to overcome this noise to maximize the accuracy of leaf shape estimates. We start in this section by modeling and quantifying the measurement noise.

3.1. Noise Characterization

Since the depth camera returns an IR reflectance in addition to a depth value at each pixel, both it and the color camera are initially calibrated using Zhang's method [14] to obtain intrinsic and extrinsic parameters. Thus each pixel in each camera defines a ray from its camera origin. Depth noise is modeled as a one dimensional random variable, ε, along the ray for each pixel along its ray direction.

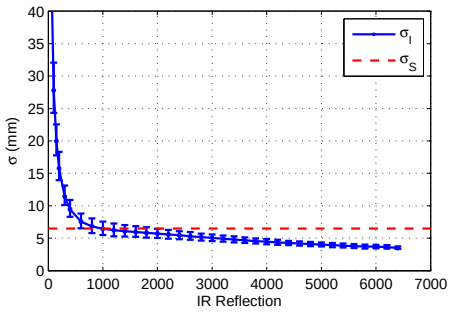

Figure 2. Image-varying noise, σ_I, is predicted well by the IR reflectance in raw units returned by the camera, see Figure 1(b). The scene-varying noise, σ_S, is plotted for comparison.

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.

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 can filter these points by discarding depth pixels with $\sigma_I > 20$mm.

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

4.2. Facet Model

Mesh fitting for an individual facet is illustrated in Figure 3. 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.

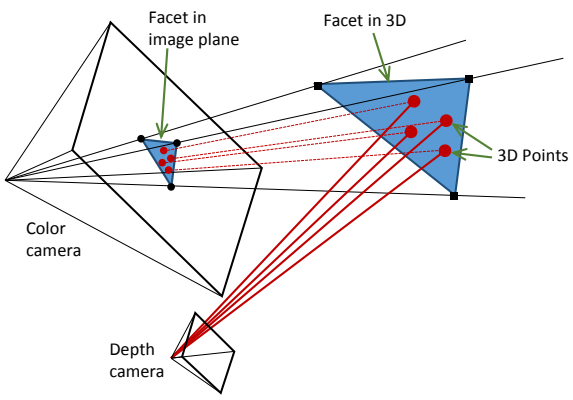

Figure 3. The parallel and adjacent color and depth cameras are shown as pyramids denoting their fields of view, and their size difference illustrates their relative resolutions. Three vertices in a color image define the rays on which the vertices of the corresponding 3D object facet must lie. This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

The next step is to associate depth measurements with the facet. Pixels in the depth camera are projected along their rays out into 3D, and then they are projected into the color image. We associate the depth pixel, $\boldsymbol{p}_i$, with the facet, $\mathcal{F}_i$, into which it projects in the color image, as illustrated in Figure 3.

To estimate the facet parameters from depth measurements we will express the depth points, $\boldsymbol{p}_i$, after they have been projected out and transformed into color camera coordinates, as a linear function of the vertices of its facet:

$$\boldsymbol{p}_i = \sum_{j \in \mathcal{F}_i} \alpha_j \boldsymbol{v}_j. \tag{3}$$

Here $\mathcal{F}$ is the set of three vertex indices belonging to the facet, and α_j is the coefficient of vertex $\boldsymbol{v}_j$ as illustrated in Figure 4, and $\sum_{j \in \mathcal{F}} \alpha_j = 1$. This linear sum is valid if we make a local orthographic approximation for the projection of a facet. It 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. Substituting in depth-scaled homogeneous vectors, and taking the third row, we obtain an equation for the point depth, λ_i, in the color image:

$$\lambda_i = \sum_{j \in \mathcal{F}_i} \alpha_j \lambda_j. \tag{4}$$

4.3. Least Squares Depth

Equation (4) gives the modeled depth of a point in terms of its facet vertices. The depth camera will provide measured depths for each pixel, indicated as $\bar{\lambda}_i$, along with a standard deviation estimate σ_i obtained from Eq. (2). Now

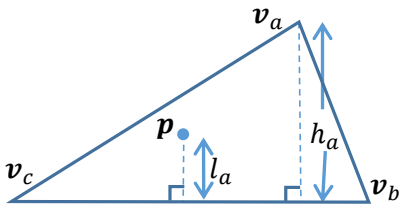

Figure 4. The coordinates of a point on a facet described by Eq. (3) are the weighted linear sum of the three vertex coordinates. The weight, α_a, for vertex $\boldsymbol{v}_a$ is given by $\alpha_a = \frac{l_a}{h_a}$, the ratio of its perpendicular distance l_a to the opposite edge to the vertex perpendicular distance h_a. Analogous expressions describe α_b and α_c.

this noise is along the depth ray, rather than the color camera ray, but since the depth and color cameras are close together compared to the distance to target, these rays are close to parallel and we assume σ_i is a good measure along the camera ray. With this approximation, the weighted least squares cost for between measured and predicted depth is:

$$E_{depth}(\Lambda_{\boldsymbol{v}}) = \sum_{i \in \mathcal{D}} \|\bar{\lambda}_i - \sum_{j \in \mathcal{F}_i} \alpha_j \lambda_j\|^2_{\sigma_i^2}, \qquad (5)$$

where $\mathcal{D}$ is the set of depth pixels that project onto the target. The norm is weighted with the inverse variance of each point. Minimizing this for $\Lambda_{\boldsymbol{v}}$, the set of all vertex depths, is a straightforward linear calculation. It estimates the mesh and it correctly minimizes the measurement error.

4.4. Shape from Shading

The depth camera measures IR reflectance in addition to depth at each pixel. Since the reflectance depends on the angle between the surface normal and the incident ray from the IR illuminator, the reflectance image can provide useful cues on the object surface. Shape from shading techniques model this dependence on the surface normal, along with additional surface assumptions such as smoothness, to estimate the normals and integrate an object surface [REF]. However the real world practicality of these methods has been limited since they generally require a single known light source position illuminating a Lambertian surface, the integration is sensitive to noise, and shape is obtained only up to a scale factor. Fortunately our application satisfies the key requirements of Shape from Shading, (we have a known light source and leafs are modeled well as Lambertian surfaces [1]), and our mesh model provides additional information that removes the need to integrate noisy surface normals. This section describes our use of Shape from Shading to improve shape recovery.

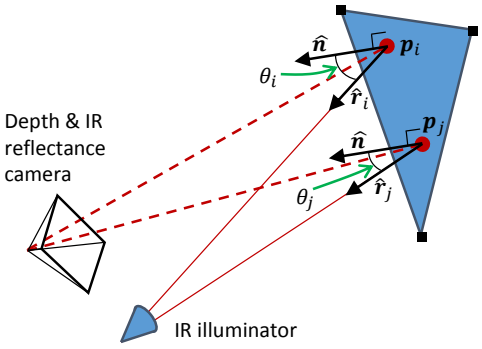

Figure 5. Our geometric model for the reflectance image. At each depth point the intensity of the reflectance image will depend on a number of factors including the angle between the facet normal and the ray to the illuminator shown for two points as θ_a and θ_b.

4.4.1 Reflectance Modeling

We build a simplified bidirectional reflectance model to explain the pixel values, R_i, of the IR reflectance image, shown in Figure 1(b). This model for pixel i is:

$$R_i = \frac{I_i \rho \hat{\boldsymbol{r}}_i \cdot \hat{\boldsymbol{n}}_i s_i}{r_i^2}. \qquad (6)$$

Here I_i is the intensity of the ray from the IR illuminator assumed to be a point source and decreasing with the inverse square of the distance to target, r. Under a Lambertian assumption the reflected beam is decreased by the albedo, ρ, and the inner product of the ray direction $\hat{\boldsymbol{r}}_i$ and the normal, $\hat{\boldsymbol{n}}_i$. Finally the sensor scales the incoming beam with a factor s_i. This model can be simplified further by approximating the outgoing ray I_i as being the same for a given pixel regardless of target depth. We define a pixel gain $g_i = \log(I_i s_i)$, and the resulting model is:

$$R_i = \frac{\rho \exp(g_i) \hat{\boldsymbol{r}}_i \cdot \hat{\boldsymbol{n}}_i}{r_i^2}. \qquad (7)$$

The gain values can be calculated from a single depth image of a known surface with constant albedo up to a scale factor (of the unknown albedo). We observe large gain near the center of the image with tapering towards the edges, along with inter-pixel variations. We chose to treat the pixel variations as noise and model the gain as a polynomial in normalized pixel coordinates, u and v. That is our modeled gain is:

$$g(u, v) = \beta_1 + \beta_2 u + \beta_3 v + \beta_4 u^2 + \beta_5 v^2 \qquad (8)$$

Taking the log of Eq. (7) we can do a least squares fit of the parameters β_i and so characterize the gain. We used a flat target with constant albedo and took data at various inclinations and depths. Figure 6 shows the resulting gain model, along with data from one depth image.

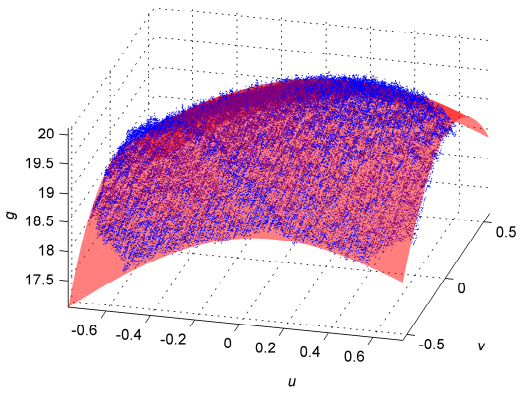

Figure 6. Our model for gain $g(u, v)$ in Eq, (8) is fit to a log reflectance image adjusted with known range and ray angles from Eq. (7). The dots are the measured data and the surface is the parameterized gain $g(u, v)$.

4.4.2 Mesh Normals Estimation

Given a reflectance image providing, R_i for each pixel, a depth image from which we can calculate range to each pixel, r_i, and our gain model, $g(u, v)$, we can rearrange and Eq. (eq:reflectance) to obtain an inclination prediction, η_i at each pixel:

$$\eta_i \equiv \rho_t \cos(\theta_i) = R_i r_i^2 \exp(-g(u, v)). \tag{9}$$

The scale factor is the unknown target albedo, ρ_t. If the target such as a leaf has a uniform albedo, this is the same for all target pixels.

Now each mesh facet has a unit normal which can be encoded in terms of the depths of its three vertices: $\hat{\boldsymbol{n}}(\lambda_a, \lambda_b, \lambda_c)$. The inner product of this normal with the unit ray to the illuminator is $\cos(\theta)$ and should agree with the values predicted in Eq. (9) for each reflectance pixel it contains. This leads to a Shape from Shading cost of

$$E_{SfS}(\boldsymbol{\lambda}_v, \rho_t) = \sum_{i \in \mathcal{T}} \| \frac{\eta_i}{\rho_t} - \hat{\boldsymbol{n}}_i \cdot \hat{\boldsymbol{r}}_i \|^2. \tag{10}$$

The sum is over all target pixels, $\mathcal{T}$, projecting into the image mesh, and $\hat{\boldsymbol{n}}_i$ indicates the normal of the facet into whose image the depth pixel projects. This cost introduces the unknown target albedo, ρ_t, as an addition parameter to be optimized over. The cost is nonlinear, but given a good initial parameters from the least squares solution to Eq. (5), it is readily minimized as a function of vertex depths and albedo.

4.5. 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 [6] and can be minimized using the discrete Laplacian operator. Here we use Laplacian smoothing due to its simplicity and good performance [6, 10, 2], although we modify it to accommodate image-based edge information. In addition, rather than apply Laplacian smoothing after the fact, entailing an iterative optimization [6], we show that Laplacian smoothing can be incorporated directly into the least squares mesh estimation. This has a number of advantages. 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 least squares solution to the Laplacian avoiding the need to iterate as in methods such as [6]. Finally when the regularization components are added to Eq. (5) they ensure that the solution is well-posed even when some of the facets have no depth points in them.

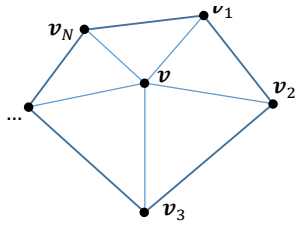

Figure 7. In discrete form the Laplacing is implemented as an umbrella operator, Eq. (11), over a vertex $\boldsymbol{v}$ and its first neighbors.

Laplacian smoothing uses an umbrella-operator [6] on a vertex, $\boldsymbol{v}$, and its neighbors $\boldsymbol{v}_i \in \mathcal{N}(\boldsymbol{v})$,

$$\boldsymbol{u}(\boldsymbol{v}) = \frac{1}{n} \sum_{\boldsymbol{v}_i \in \mathcal{N}} \boldsymbol{v}_i - \boldsymbol{v}, \tag{11}$$

for n neighbors as illustrated in Figure 7. In our model the vertices lie along known rays and so this operator can be expressed as a function of the vertex depth: $\boldsymbol{u}(\lambda) = \frac{1}{n} \sum_{i \in \mathcal{N}} \lambda_i \tilde{\boldsymbol{v}}_i - \lambda \tilde{\boldsymbol{v}}$. 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 $\tilde{\boldsymbol{v}}$ components into a single matrix U, we obtain

$$E_{reg}(\boldsymbol{\lambda}_v) = \|U\boldsymbol{\lambda}\|^2. \tag{12}$$

The cost in Eq. (12) penalizes high curvature regions, and so provides a way incorporate smoothness priors into the mesh estimation. Now we may have image cues for creases or sharp edges on portions of a mesh, as we describe in section 5. We can modify the umbrella operator from Eq. (11) so that the neighbors are only other vertices on the crease. In this way the Laplacian acts along the crease, and not across it, allowing sharper folding along these edges.

5. Color-Based Mesh Cues

Since we do not deal with overlapping and highly cluttered background, the segmentation can be done easily by using the color information of the rgb image. The initial segmentation on the leaf is done by using the K-means cluster on the a and b channels of the Lab color space. It is found that the a and b channels contain much better information about the green pigments of the leaf. Using only three clusters on these two channels in the k-means clustering, it is able to segment out the leaves from the background. Mask is built based on the segmented plant imagery and the boundary of the leaf is traced based on the developed mask.

We build line segment based on boundary pixels of the plant leaf boundaries. Since plant leaves are clumped together in a single structure, it is possible to get closed boundary on the entire segmented image of the plant. We then perform a polygonal approximation motivated by the merging technique [7] process on the boundary by approximating straight lines with a maximum deviation of two pixels from pixels in the original boundary. The two end points of the approximated lines are saved as vertices that join an edge of the polygon.

Having obtained the polygonal approximation of the plant leaf boundaries, we then sample points with a uniform spacing of ℓ pixels on each side of the polygon. Uniform grid of points with a spacing of r are also created in the entire image, and the grid points which fall within or on the mask structure are only selected. Points falling within $\ell \leq r$ are removed from the set of selected grid points. Both the selected grid points and the sampled points on the boundary are then used to create a delaunay triangulation on the plant mask.The slivers of triangular facets that lie outside the polygonal shape are removed by seeking whether the midpoints of the line fall within the polygonal shape or not.The final meshes on a typical rgb plant image looks like in Figure 8:

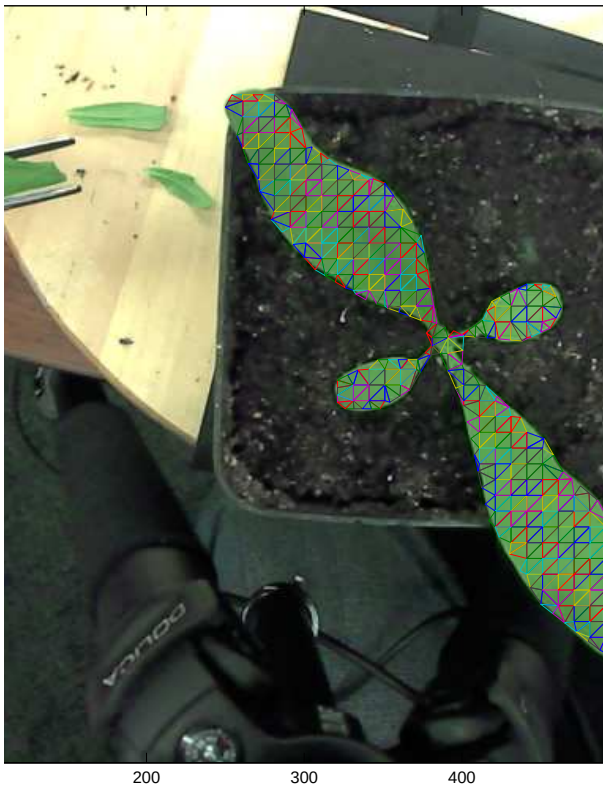

Figure 8. Mesh Distribution on the RGB Plant Imagery

6. Results

7. Conclusion

References

[1] M. Chelle. Could plant leaves be treated as lambertian surfaces in dense crop canopies to estimate light absorption? *Ecological Modelling*, 198(1):219 – 228, 2006. 4

[2] C.-Y. Chen and K.-Y. Cheng. A sharpness dependent filter for mesh smoothing. *Computer Aided Geometric Design*, 22(5):376 – 391, 2005. Geometry Processing. 5

[3] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 303–312, New York, NY, USA, 1996. ACM. 1

[4] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pages 295–302, New York, NY, USA, 1994. ACM. 1, 2

[5] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proc. of ACM UIST*, pages 559–568, 2011. 1

[6] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 105–114, New York, NY, USA, 1998. ACM. 1, 5

[7] J.-G. Leu and L. Chen. Polygonal approximation of 2-d shapes through boundary merging. *Pattern Recognition Letters*, 7(4):231 – 238, 1988. 6

[8] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. W. Fitzgibbon. Kinectfusion: Real-time dense surface

mapping and tracking. In *Proc. of IEEE ISMAR*, pages 127–136, 2011. 1

[9] 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. 2

[10] Y. Ohtake, A. Belyaev, and I. Bogaevski. Mesh regularization and adaptive smoothing. *Computer-Aided Design*, 33(11):789 – 800, 2001. 5

[11] J. Sienz, I. Szarvasy, E. Hinton, and M. Andrade. Computational modelling of 3d objects by using fitting techniques and subsequent mesh generation. *Computers & Structures*, 78(13):397 – 413, 2000. 1

[12] G. Turk and M. Levoy. Zippered polygon meshes from range images. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pages 311–318, New York, NY, USA, 1994. ACM. 1

[13] I.-C. Yeh, C.-H. Lin, O. Sorkine, and T.-Y. Lee. Template-based 3d model fitting using dual-domain relaxation. *Visualization and Computer Graphics, IEEE Transactions on*, 17(8):1178–1190, Aug 2011. 1

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