Contour Detection in Multi-Angle Time-Lapse Images of Growing Plants

Merel te Hofsté

TU Delft

Contour Detection in Multi-Angle Time-Lapse Images of Growing Plants

by

Merel te Hofsté

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday December 17, 2020 at 10:00 AM.

Student number:	4213157	
Project duration:	March 1, 2020 – December 17, 2020	
Thesis committee:	Prof. dr. ir. C. Vuik,	TU Delft, Chair
	Dr. N. V. Budko,	TU Delft, Supervisor
	Dr. ir. R. F. Remis,	TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Contents

1

Introduction

Mathematics applied to agricultural problems can lead to insightful observations that can help to improve harvest becoming more efficient. HZPC is a Dutch company specialized in seed potatoes which distributes these seeds all over the world. Their goal is to develop seeds suitable for different climates and cultures. To this end, data has been collected in the form of images of plants of different breeds and varieties in multiple climate rooms over time. This way, the growth of a plant is visualized.

Figure 1.1: Images of a plant over time.

The problem of quantifying the growth dynamics of plants from time-lapse images is important for determining variety-specific characteristics and subsequent breeding. Here, mathematical models and imaging techniques can contribute to this problem. In the current setup, a camera is moving along a table in every climate room as shown in Figure 1.2. This results in a minimum of three images of every plant, breed and variety, taken from different angles. With multiple images one can obtain information about the depth in the image, i.e. a 3D reconstruction can be made.

In the first Chapter we will explore the Snakes algorithm that detects the contour of the leaves of the plant in combination with a Gradient Vector Field.

In the second Chapter a pinhole camera model is explained. Using the three two-dimensional contours found in the first Chapter, we will develop an algorithm that finds point correspondences on these contours, estimates the so-called Essential matrix, and, employing the projection equations of the pinhole camera model, reconstructs the three-dimensional contour of the plant's canopy edge.

In the final Chapter we will theoretically explore the possibility to combine the contour detection with the point matching algorithm and estimation of the Essential matrix which we will introduce as the Double Snakes method.

Figure 1.2: Set-up of moving camera above table.

2

Outline detection

In this Chapter the Traditional Snakes method is introduced and extended with the Gradient Vector Flow field. After some pre-processing and placing the initial contour, the Snakes-GVF algorithm is applied on images taken under different angles.

2.1. Traditional Snake

The energy functional of a traditional snake [1] parameterized by $\mathbf{q}(s) = [u(s), v(s)]$, with u and v the pixel coordinates of an image, can be written as:

$$\begin{aligned} E^*_{\text{snake}} &= \int_0^1 E_{\text{snake}}\left(\mathbf{q}(s)\right) \mathrm{d}s \\ &= \int_0^1 E_{\text{int}}\left(\mathbf{q}(s)\right) + E_{\text{ext}}(\mathbf{q}(s)) \,\mathrm{d}s, \end{aligned} \tag{2.1}$$

where E_{int} contains the internal energy of the spline due to bending and E_{ext} contains the external constraint forces. The internal energy of the snake consists of the continuity of the contour E_{cont} and the smoothness of the contour E_{curv},

$$E_{\text{int}} = E_{\text{cont}} + E_{\text{curv}}.$$

Including the weights α en β, which control the amount of stretch and curvature in the snake respectively, we can expand the energy functional as

$$E[\mathbf{q}] = \int_0^1 \frac{1}{2}\left(\alpha \left\|\mathbf{q}'(s)\right\|^2 + \beta \left\|\mathbf{q}''(s)\right\|^2\right) + E_{\text{ext}}\left(\mathbf{q}(s)\right) \,\mathrm{d}s. \tag{2.2}$$

To derive the Euler-Lagrange equations of (2.2), we assume a local minimum of E in $\mathbf{q}$. Let $\boldsymbol{r}(s) = [\mu(s), \nu(s)]$ be a twice continuously differentiable fixed curve. Then we have $E[\mathbf{q}] \leq E[\mathbf{q} + \varepsilon\mathbf{r}]$ for any small number $\varepsilon > 0$. So the function $\Phi(\varepsilon) = E[\mathbf{q} + \varepsilon\mathbf{r}]$ is minimal for $\varepsilon = 0$. Therefore the derivative of Φ equals zero at $\varepsilon = 0$.

$$\begin{aligned} \Phi(\varepsilon) &= E[\mathbf{q} + \varepsilon\mathbf{r}] \\ &= \int_0^1 \frac{1}{2}\left(\alpha \left\|\mathbf{q}' + \varepsilon\mathbf{r}'\right\|^2 + \beta \left\|\mathbf{q}'' + \varepsilon\mathbf{r}''\right\|^2\right) + E_{\text{ext}}\left(\mathbf{q} + \varepsilon\mathbf{r}\right) \,\mathrm{d}s. \end{aligned}$$

Taking the derivative of $\Phi(\varepsilon)$ yields

$$\Phi'(\varepsilon) = \int_0^1 \alpha\left(\mathbf{q}' + \varepsilon\mathbf{r}'\right)\cdot\mathbf{r}' + \beta\left(\mathbf{q}'' + \varepsilon\mathbf{r}''\right)\cdot\mathbf{r}'' + \nabla E_{\text{ext}}(\mathbf{q})\cdot\mathbf{r}\,\mathrm{d}s$$

Substituting $\Phi'(0) = 0$, using period boundary conditions of $\mathbf{q}', \mathbf{q}'', \mathbf{q}''', \mathbf{r}', \mathbf{r}''$ and using partial integration we get

$$\begin{aligned} \Phi'(0) &= \int_0^1 \alpha\left(\mathbf{q}'\cdot\mathbf{r}'\right) + \beta\left(\mathbf{q}''\cdot\mathbf{r}''\right) + \nabla E_{\text{ext}}(\mathbf{q})\cdot\mathbf{r}\,\mathrm{d}s \\ &= \int_0^1 \left[-\alpha\mathbf{q}'' + \beta\mathbf{q}'''' + \nabla E_{ext}(\mathbf{q})\right]\cdot\mathbf{r}\,\mathrm{d}s = 0 \end{aligned}$$

for any $\mathbf{r}(s)$. We can conclude that any curve $\mathbf{q}(s)$ that minimizes the functional E must satisfy the equation

$$\alpha\mathbf{q}'' - \beta\mathbf{q}'''' - \nabla E_{ext}(\mathbf{q}) = 0. \tag{2.3}$$

2.2. Gradient Vector Flow

Typical external energies for the active contour are

$$E^1_{\text{ext}}(\mathbf{q}) = -|\nabla I(\mathbf{q})|^2, \tag{2.4}$$

$$E^2_{\text{ext}}(\mathbf{q}) = -|\nabla\big(G_\sigma(\mathbf{q}) * I(\mathbf{q})\big)|^2, \tag{2.5}$$

where $I(\mathbf{q})$ is a given gray-scale image and $G_\sigma(\mathbf{q})$ is a two-dimensional Gaussian function with standard deviation σ. Here $*$ is the convolution operator.

The Gradient Vector Flow (GVF) [2] is an approach to define a new external force field $\mathbf{w}(u,v) = \big[\phi(u,v), \psi(u,v)\big]$, called the gradient vector flow field, which minimizes the energy functional

$$\mathscr{E}[\mathbf{w}] = \iint \mu\left(\phi_u^2 + \phi_v^2 + \psi_u^2 + \psi_v^2\right) + |\nabla f|^2|\mathbf{w} - \nabla f|^2 \,\mathrm{d}u\mathrm{d}v, \tag{2.6}$$

where μ is a controllable smoothing term and should be set according to the amount of noise in the image and f is an edge map derived from the image $I(u,v)$. The negative external energy in (2.3) can be replaced by this vector field $\mathbf{w}(u,v)$, defining a new snake called the GVF snake:

$$\alpha\mathbf{q}'' - \beta\mathbf{q}'''' + \mathbf{w} = 0. \tag{2.7}$$

The Gradient Vector Field was introduced to overcome flaws in the typical external energies, which we will discuss later.
In order to find the vector field $\mathbf{w}$ to minimize the functional in (2.6), we can derive the corresponding Euler-Lagrange equations. Therefore we assume that $\mathscr{E}$ attains a minimum in $\mathbf{w}$. Let $\boldsymbol{s}(u,v) = \big[\chi(u,v), \omega(u,v)\big]$ be a vector field and define $\Psi(\varepsilon) = \mathscr{E}[\mathbf{w} + \varepsilon\boldsymbol{s}]$ for small values $\varepsilon > 0$. The function Ψ attains its minimum at $\varepsilon = 0$ and therefore $\Psi'(0) = 0$.

$$\begin{aligned}\Psi(\varepsilon) &= \mathscr{E}[\mathbf{w} + \varepsilon\boldsymbol{s}] \\ &= \iint \mu\left((\phi_u + \varepsilon\chi_u)^2 + (\phi_v + \varepsilon\chi_v)^2 + (\psi_u + \varepsilon\omega_u)^2 + (\psi_v + \varepsilon\omega_v)^2\right) \\ &\quad + |\nabla f|^2|\mathbf{w} + \varepsilon\boldsymbol{s} - \nabla f|^2 \,\mathrm{d}u\mathrm{d}v.\end{aligned}$$

The derivative of Ψ with respect to ε is given by

$$\begin{aligned}\Psi'(\varepsilon) = 2\iint &\mu\left((\phi_x + \varepsilon\chi u)\chi_u + (\phi_v + \varepsilon\chi_v)\chi_v + (\psi_u + \varepsilon\chi_u)\chi_u + (\psi_v + \varepsilon\omega_v)\omega_v\right) \\ &+ |\nabla f|^2\left(\mathbf{w} + \varepsilon\boldsymbol{s} - \nabla f\right)\cdot\boldsymbol{s} \,\mathrm{d}u\mathrm{d}v.\end{aligned}$$

Substituting $\varepsilon = 0$ yields

$$\begin{aligned}\Psi'(0) &= 2\iint \mu\left(\phi_u\chi_u + \phi_v\chi_v + \psi_u\omega_u + \psi_v\omega_v\right) \\ &\quad + |\nabla f|^2\left(\mathbf{w} - \nabla f\right)\cdot\boldsymbol{s} \,\mathrm{d}u\mathrm{d}v \\ &= 2\iint \mu\left(\nabla\phi\cdot\nabla\chi + \nabla\psi\cdot\nabla\omega\right) + |\nabla f|^2\left(\mathbf{w} - \nabla f\right)\cdot\boldsymbol{s} \,\mathrm{d}u\mathrm{d}v.\end{aligned} \tag{2.8}$$

We can simplify the integral in (2.8) by using integration by parts followed by Gauss' theorem or use the first identity of Green immediately.

$$\begin{aligned}\iint_\Omega \nabla\phi\cdot\nabla\chi \,\mathrm{d}u\mathrm{d}v &= \iint_\Omega \nabla\cdot\left(\chi\nabla\phi\right) - \chi(\nabla\cdot\nabla\phi) \,\mathrm{d}u\mathrm{d}v \\ &= \int_{\partial\Omega} \chi\nabla\phi\cdot\mathbf{n} \,\mathrm{d}\Gamma - \iint_\Omega \chi\nabla^2\phi \,\mathrm{d}u\mathrm{d}v \\ \iint_\Omega \nabla\psi\cdot\nabla\omega \,\mathrm{d}u\mathrm{d}v &= \iint_\Omega \nabla\cdot\left(\omega\nabla\psi\right) - \omega(\nabla\cdot\nabla\psi) \,\mathrm{d}u\mathrm{d}v \\ &= \int_{\partial\Omega} \omega\nabla\psi\cdot\mathbf{n} \,\mathrm{d}\Gamma - \iint_\Omega \omega\nabla^2\psi \,\mathrm{d}u\mathrm{d}v\end{aligned}$$

In order to get rid of the integral concerning the boundary, we need either $\boldsymbol{s}$ or $\nabla\phi \cdot \mathbf{n}$ and $\nabla\psi \cdot \mathbf{n}$ to vanish on $\partial\Omega$. We impose $\frac{\partial\phi}{\partial\mathbf{n}} = \frac{\partial\psi}{\partial\mathbf{n}} = 0$ on the boundary so that (2.8) becomes

$$\Psi'(0) = -2\iint_{\Omega} \mu\left(\nabla^2\mathbf{w} + |\nabla f|^2\left(\mathbf{w} - \nabla f\right)\right) \cdot \boldsymbol{s}\,\mathrm{d}u\mathrm{d}v = 0 \tag{2.9}$$

for any vector field $\boldsymbol{s}$. Concluding that any vector field $\mathbf{w} = (\phi, \psi)$ that minimizes the functional $\mathscr{E}$ must satisfy the equations

$$\mu\nabla^2\phi - \left(f_u^2 + f_v^2\right)\left(\phi - f_u\right) = 0, \tag{2.10}$$
$$\mu\nabla^2\psi - \left(f_u^2 + f_v^2\right)\left(\psi - f_v\right) = 0. \tag{2.11}$$

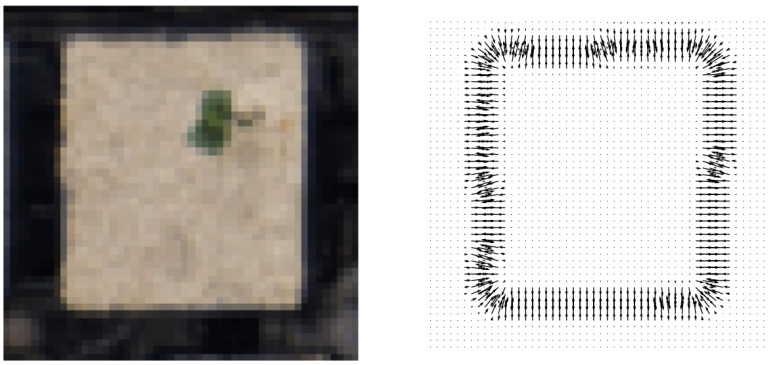
Edge map

To determine the gradient vector flow field for a certain image I, the edge map f corresponding to I must be calculated. Common choices for the edge map of gray-scale images are

$$f^1\left(\mathbf{q}\right) = |\nabla I(\mathbf{q})|^2,$$
$$f^2\left(\mathbf{q}\right) = |\nabla\left(G_\sigma(\mathbf{q}) * I(\mathbf{q})\right)|^2,$$

having the property to be larger at regions of interest, e.g. boundaries. The vector field of f is pointing towards these regions, but often these vectors have a narrow capture range. In addition, in every homogeneous region of the pre-processed image, ∇f is zero. Using the GVF field we can improve these disadvantages.

For example, we can take a reduced image of a plant's container of 50×50 pixels, where we pre-processed the image so that the plant is not of interest. The image is reduced so that the arrows of the vector field are clearly visible. The vector field of edge map f^2 with $\sigma = 1.1$ is shown in Figure 2.1.



(a) Original image

(b) Vector field of edge map f of box.

Figure 2.1: Vector field of edge map f after the original image is pre-processed to remove the plant. (reduced image)

2.3. Discretization

In order to solve the GVF snake equation (2.7), it is made dynamic by setting $\mathbf{q}(s)$ time-dependent as well, i.e. $\mathbf{q}(s,t)$. The partial derivative $\frac{\partial\mathbf{q}}{\partial t}$ is set equal to the left hand side of (2.7) to obtain the dynamic GVF snake equation

$$\frac{\partial\mathbf{q}}{\partial t} = \alpha\frac{\partial^2\mathbf{q}}{\partial s^2} - \beta\frac{\partial^4\mathbf{q}}{\partial s^4} + \mathbf{w}(\mathbf{q}). \tag{2.12}$$

We can solve (2.12) by discretization and solving the discrete system iteratively. Therefore, we first have to compute $\mathbf{w}$.

GVF field

To compute the GVF field $\mathbf{w}(\mathbf{q})$ we can solve the equations (2.10) and (2.11) numerically. We will illustrate the numeric approximation of (2.10), whereas (2.11) will follow a similar approach. Consider the boundary value problem

$$\begin{cases} \mu\nabla^2\phi - g\psi &= -g f_u, \quad \text{in } \Omega \\ \frac{\partial\phi}{\partial n} &= 0, \quad \text{on } \partial\Omega \end{cases}, \quad \text{where } g = f_u^2 + f_v^2 \tag{2.13}$$

where Ω is the domain of the image $I(u,v)$. We divide the domain Ω in $n \times m$ small rectangles according to the pixels of the image. Let $\phi_{i,j}$ be the value of ϕ at pixel (i,j) for $i = 0,\ldots,n-1$ in the vertical direction and $j = 0,\ldots,m-1$ in the horizontal direction. Note that we can set the distance between these pixels at one, i.e. $\Delta u = \Delta v = 1$. Using central difference for the Laplacian term in u and v we obtain the following equations

$$\left(\frac{\partial^2\phi}{\partial u^2}\right)_{i,j} \approx \phi_{i-1,j} - 2\phi_{ij} + \phi_{i+1,j},$$
$$\left(\frac{\partial^2\phi}{\partial v^2}\right)_{i,j} \approx \phi_{i,j-1} - 2\phi_{ij} + \phi_{i,j+1}.$$

For pixels at the boundary, we use the Neumann boundary condition for the outward pointing normal vector $\mathbf{n} = (1,0)^T$ and $\mathbf{n} = (0,1)^T$ for u and v respectively, approximated with central differences.

$$\left(\frac{\partial\phi}{\partial u}\right)_{0,j} \approx \phi_{-1,j} - \phi_{1,j} = 0 \implies \left(\frac{\partial^2\phi}{\partial u^2}\right)_{0,j} \approx -\phi_{0,j} + \phi_{1,j}$$
$$\left(\frac{\partial\phi}{\partial u}\right)_{n-1,j} \approx \phi_{n-2,j} - \phi_{n,j} = 0 \implies \left(\frac{\partial^2\phi}{\partial u^2}\right)_{n,j} \approx \phi_{n-2,j} - \phi_{n-1,j}$$

We can substitute this result to obtain the following matrices.

$$L_{uu} = \begin{bmatrix} -1 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -1 \end{bmatrix}, \quad L_{vv} = \begin{bmatrix} -1 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -1 \end{bmatrix}$$

We need the $n \times n$ matrix L_{uu} to discretize the domain for every column j and the $m \times m$ matrix L_{vv} to discretize the domain for every row i. In Kronecker product notation this becomes

$$L = I_{\text{col}} \otimes L_{uu} + L_{vv} \otimes I_{\text{row}}, \tag{2.14}$$

where I_{col} is a $m \times m$ identity matrix and I_{row} is a $n \times n$ identity matrix corresponding with the number of columns and rows of the pixels in the domain. Now, L becomes a $nm \times nm$ matrix.

Besides the Laplacian in (2.13), the derivative of the edge map f with respect to u and v has to be approximated. We will again use central differences, except for pixels at the boundary, which we will approximate with a forward or backward difference. Let $f_{i,j}$ be the value of the edge map at pixel (i,j) in the same order as $\phi_{i,j}$. Defining a matrix F containing all the values of the edge map f, i.e. $F_{i,j} = f_{i,j}$, we can define the matrices F^u and F^v containing all the element-wise derivatives of F with respect to u and v.

$$F_u \approx C_u F,$$
$$F_v \approx (C_v F^T)^T = F C_v^T,$$

where

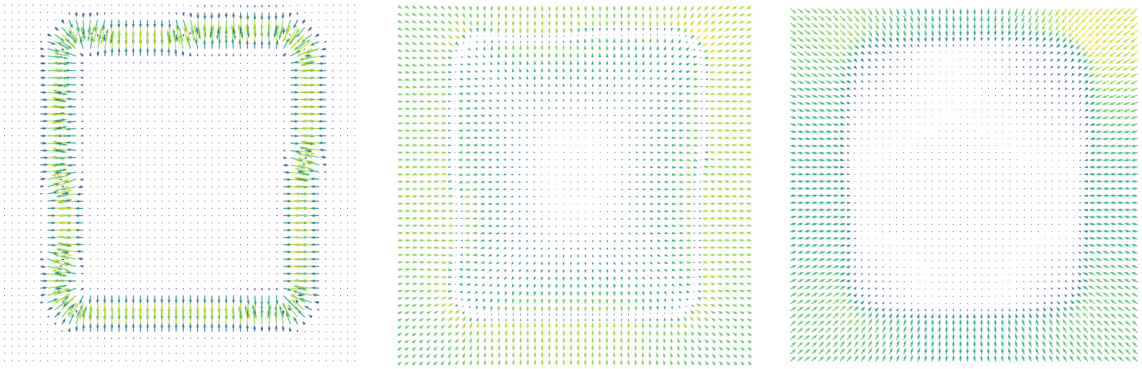
$$C_u = \begin{bmatrix} -1 & 1 & & & \\ -\frac{1}{2} & 0 & \frac{1}{2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{1}{2} & 0 & \frac{1}{2} \\ & & & -1 & 1 \end{bmatrix}, \quad C_v = \begin{bmatrix} -1 & 1 & & & \\ -\frac{1}{2} & 0 & \frac{1}{2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{1}{2} & 0 & \frac{1}{2} \\ & & & -1 & 1 \end{bmatrix}.$$

Now, the discretized version of (2.13) becomes

$$\begin{aligned}(\mu L - G)\tilde{\boldsymbol{\phi}} &= -G\mathbf{f}_u \\ (\mu L - G)\tilde{\boldsymbol{\psi}} &= -G\mathbf{f}_u\end{aligned} \tag{2.15}$$

where $\mathbf{f_u}$ is the matrix F_u flattened (or vectorized) to obtain the dimension $nm \times 1$ and $G = \text{diag}(\mathbf{f_u}^2 + \mathbf{f_v}^2)$, with $\mathbf{f}^2$ the pointwise power operation.

For the system (2.15) multiple linear solvers can be used. For now, due to performance explored in [3], the Minimal Residual Method (MINRES) is used. The parameter μ is a controllable smoothing term. For $\mu = 0$ the functional (2.6) is minimized for $\mathbf{w} = \mathbf{f}$ and we will get the same results as in Figure 2.1. For a larger μ, the vector field will get too smooth and will point in the same direction.



(a) $\mu = 0$. (b) $\mu = 1\text{e}{-3}$. (c) , $\mu = 0.1$.

Figure 2.2: GVF field of the box of a plant for different μ.

Dynamic snake equation

Secondly, the dynamic GVF snake equation (2.12) has to be discretized. We want to discretize the curve $\mathbf{x}(s,t)$ by N equidistant gridpoints for s so that $\Delta s = 1$, i.e. $s_i = i$ for $i = 0,\ldots,N-1$. Now we can approximate the partial derivatives with respect to s with central differences.

$$\frac{\partial^2 \boldsymbol{q}}{\partial s^2} \approx \boldsymbol{q}_{i-1} - 2\boldsymbol{q}_i + \boldsymbol{q}_{i+1}, \quad \frac{\partial^4 \boldsymbol{q}}{\partial s^4} \approx \boldsymbol{q}_{i-2} - 4\boldsymbol{q}_{i-1} + 6\boldsymbol{q}_i - 4\boldsymbol{q}_{i+1} + \boldsymbol{q}_{i+2}$$

Using the periodicity of the curve $\mathbf{q}(s)$ we can define the following matrices.

$$D_2 = \begin{bmatrix} -2 & 1 & & & & 1 \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ 1 & & & & 1 & -2 \end{bmatrix}, \quad D_4 = \begin{bmatrix} 6 & -4 & 1 & & & 1 & -4 \\ -4 & 6 & -4 & 1 & & & 1 \\ 1 & -4 & 6 & -4 & 1 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & 1 & -4 & 6 & -4 & 1 \\ 1 & & & 1 & -4 & 6 & -4 \\ -4 & 1 & & & 1 & -4 & 6 \end{bmatrix}.$$

With Euler forward as the time integration method, we obtain the following discrete system of the dynamic snake equation (2.12).

$$\begin{aligned}\mathbf{u}_{k+1} &= \mathbf{u}_k + \Delta t\left(\alpha D_2 - \beta D_4\right)\mathbf{u}_k + \Delta t\boldsymbol{\phi}_k \\ \mathbf{v}_{k+1} &= \mathbf{v}_k + \Delta t\left(\alpha D_2 - \beta D_4\right)\mathbf{v}_k + \Delta t\boldsymbol{\psi}_k\end{aligned} \tag{2.16}$$

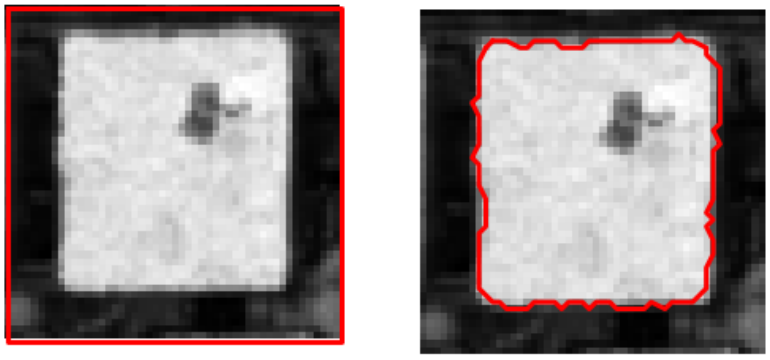
Before we solve the system (2.16), we introduce a new parameter γ controlling the strength of the GVF field. Due to the relatively low magnitude of the GVF field compared to the internal forces, we want to scale the

forces to be in the same order of magnitude. Therefore we set γ equal to the inverse of the average value of the GVF field, e.g. $\boldsymbol{\gamma} = \overline{|\mathbf{v}|}^{-1}$. The discrete system of (2.12) becomes

$$\begin{aligned} \mathbf{u}_{k+1} &= \mathbf{u}_k + \Delta t \left((\alpha D_2 - \beta D_4) \mathbf{u}_k + \gamma_u \boldsymbol{\phi}_k \right) \\ \mathbf{v}_{k+1} &= \mathbf{v}_k + \Delta t \left((\alpha D_2 - \beta D_4) \mathbf{v}_k + \gamma_v \boldsymbol{\psi}_k \right) \end{aligned} \tag{2.17}$$

Results

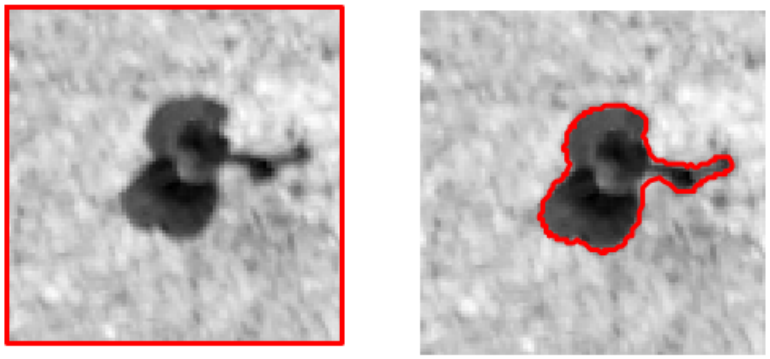
We can solve (2.17) for the reduced image of 50×50 pixels where the image is pre-processed so that the plant is not of interest. By placing an initial square at the edge of the image of $n = 100$ points, setting $\alpha = 1, \beta = 0, \Delta t = 0.01, \mu = 1\text{e}{-3}$, we see the initial curve follows the direction of the GVF field in Figure 2.2b and moves towards the boundary of the container. Note that the final curve in Figure 2.3 is not that smooth, because of the low resolution of the image.



(a) Initial curve

(b) Final curve at $t = 15$

Figure 2.3: GVF snake applied at box container after the original image is pre-processed to remove the plant. (reduced image).



(a) Initial curve

(b) Final curve at $t = 15$

Figure 2.4: GVF snake applied at contour outline of plant (reduced image).

2.4. Pre-processing

In order to apply the GVF-Snakes algorithm on the plant's outline, we have to pre-process every image and place an initial curve that is attracted to the plant's outline and not attracted to the plant's container, despite the size or location of the plant. To do this, we have the following options:

- First locate the box from outside, then crop the image and again apply GVF-Snakes to locate the plant. *This approach is feasible, but applying the algorithm twice is computationally expensive.*

- Locate the plant and place a small initial curve in the center of the plant and locate the plant's outline from inside. *Because the region of interest compared to the full image is often really small, it is harder to smooth the GVF field nicely inside the plant's outline.*
- Locate the plant and place an initial curve around it and locate the plant's outline from outside. *We will continue with this approach.*

Color filter

To locate the plant, we will make use of a green color filter. A colored image is often represented in the commonly known RGB (Red Green Blue) color model. Another representation of this RGB color model is HSV (Hue Saturation Value). HSV is designed by computer graphics researchers in 1970 and it is more convenient to use for color filtering.

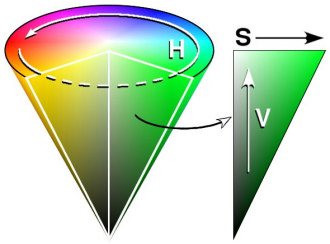
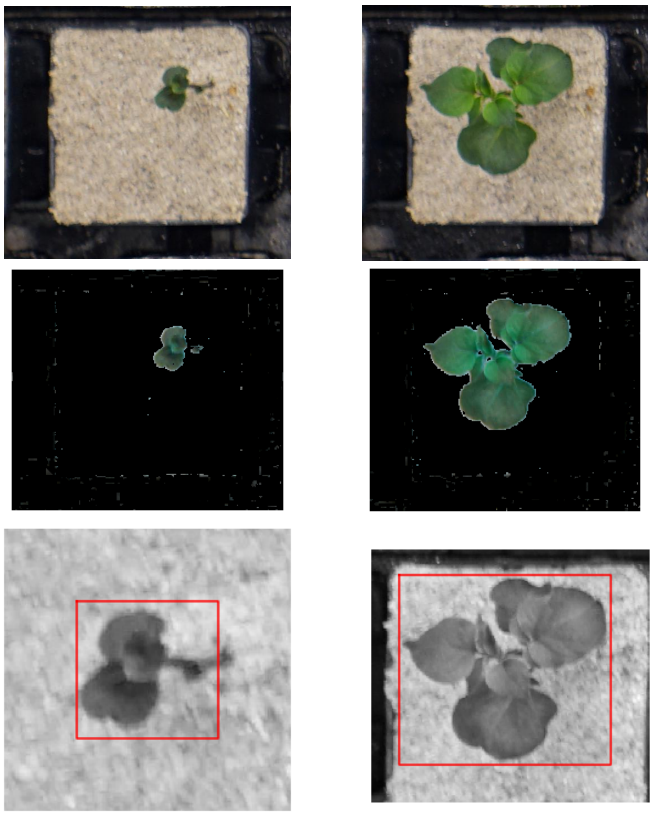


Figure 2.5: HSV color cone.

Using a range of Hue between 25 and 90 and Saturation and Value between 40 and 240 we get the results shown in Figure 2.6. Using a wider range will include more colors, with the consequence of more noise in the image. Noise reduction can be performed by removing small objects in the image and other image filters.



(a) Snapshot 6

(b) Snapshot 10

Figure 2.6: Green detection and initial curves.

Once the green in every image is detected, we can place a box around this area and use it as the initial curve placed in the original image as shown in Figure 2.6.
To compute the GVF field of the original images, some pre-processing has to be done. All images are converted to grayscale with pixel values between 0 (black) and 1 (white). Because we are interested in the darker regions of the image, we copy the image for values below a certain threshold θ to obtain a binary image. We will use $\theta = 0.5$ in the following pages.

Figure 2.7: Pre-processed image with $\theta = 0.5$.

As the plant gets bigger, it is more likely to come closer to the container. In order for the snake to still approach the plant's outline instead of the container, we can use the green mask as an extra push p 'towards' the plants' outline. This push will create more contrast in the pre-processed image, such that the edge map, and therefore the GVF field, will be higher at the green regions.

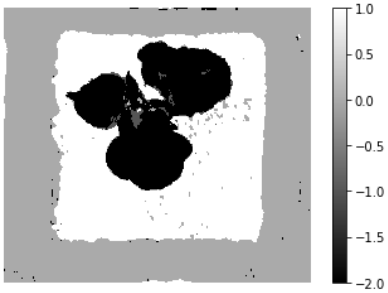
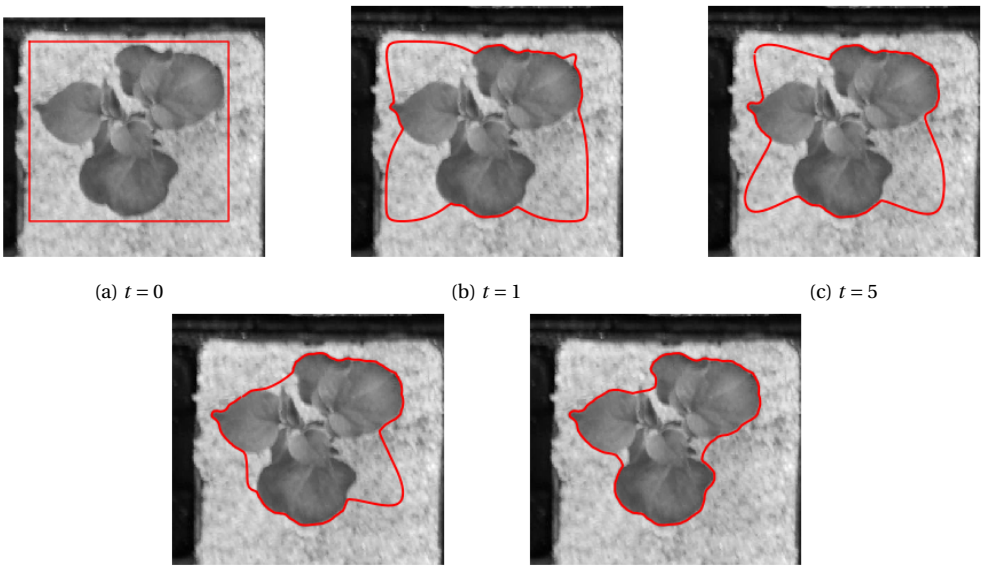


Figure 2.8: Pre-processed image with $\theta = 0.5$ and $p = 2$.

2.5. Results

The final results are obtained using n=240 points with the parameters $\alpha = 1, \beta = 0, \Delta t = 0.01, \mu = 0.1$ and $\sigma = 1.3$ for the Gaussian function G_σ of the edge map $f^2(\mathbf{q})$. In Figure 2.9 the movement of the contour is shown over time. The plant's outline close to the initial contour is detected very fast. At $t = 50$ the final contour is shown which fits the outline of the plant very nicely.

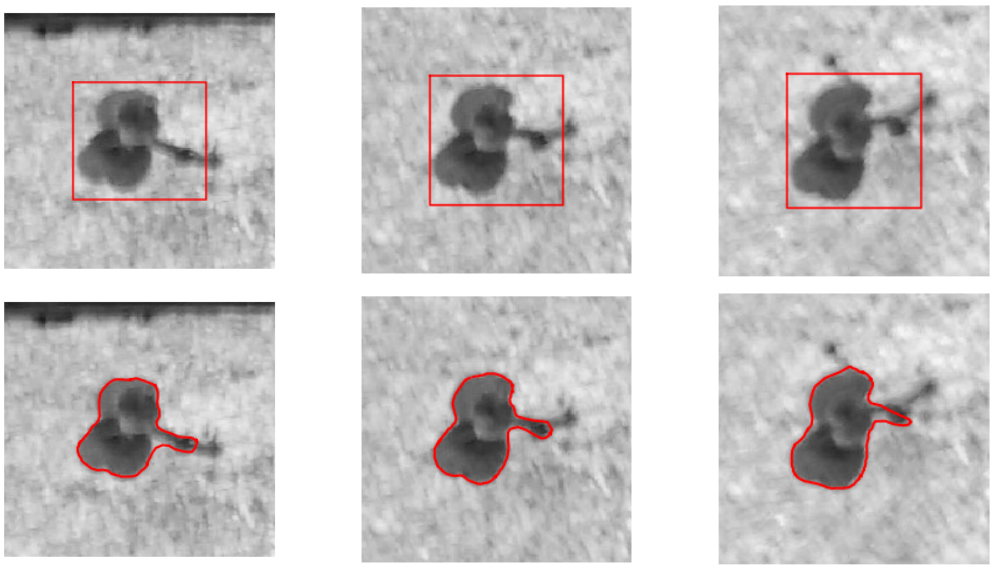


(a) $t = 0$ (b) $t = 1$ (c) $t = 5$

(d) $t = 10$ (e) $t = 50$

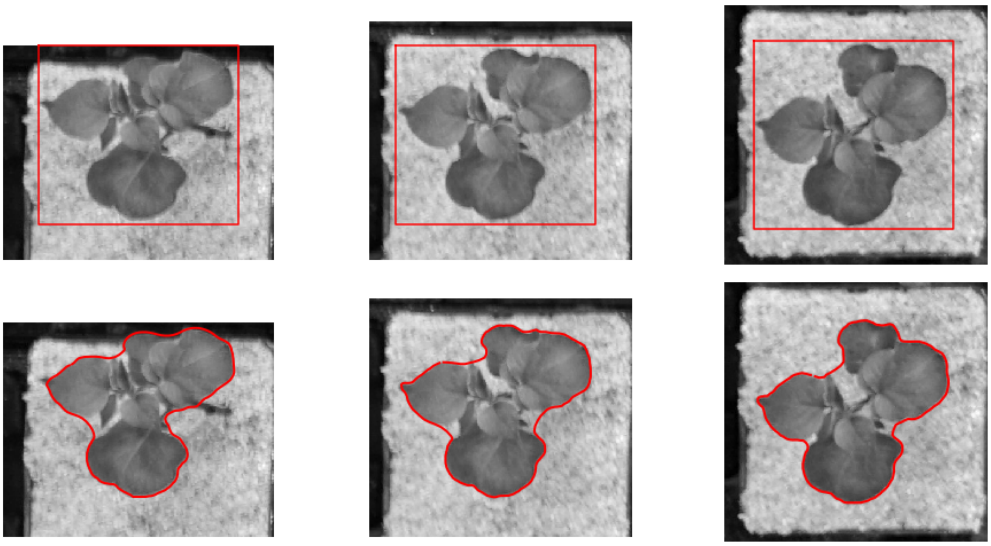
Figure 2.9: GVF snake applied on image of plant at different timesteps .

For every plant, we have 3 images taken from different angles. For every image, we can apply the algorithm to detect the outline. Because we are interested in growth, this has also been done for the same plant in a later stage. In Figure 2.10 the different contours are shown for snapshot 6 and in Figure 2.11 the same plant is shown for snapshot 10. The time interval between these snapshots is 13 days.



(a) Angle -1 (b) Angle 0 (c) Angle +1

Figure 2.10: GVF snake applied on images of plant from different angles for snapshot 6.



(a) Angle -1 (b) Angle 0 (c) Angle +1

Figure 2.11: GVF snake applied on images of plant from different angles for snapshot 10.

3

Three-dimensional contour reconstruction

In this Chapter the standard methods of photogrammetry are applied to reconstruct the three-dimensional image of the plant's edge based on two two-dimensional contours independently detected with the Snakes-GVF algorithm.

3.1. Pinhole camera model

A camera's functioning can be described by a, simplified, mathematical model: the pinhole camera model. A pinhole camera consists of a light-proof box, where light enters through a small hole in the front and is projected upside down on the back of the box, i.e. the image plane. For convenience, the image plane is often brought in front of the box. The focal length describes the distance from the pinhole to the (virtual) image plane.

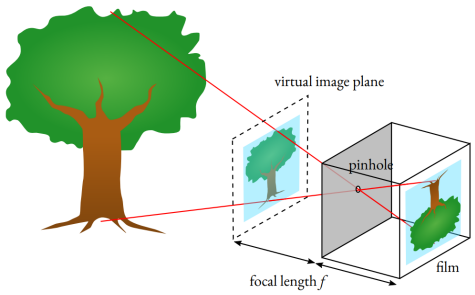


Figure 3.1: A pinhole camera model. A light-proof box where light enters via a small hole.

The pinhole camera model projects an object point in camera coordinates $\mathbf{r}' = [x', y', z']$ onto the image plane with coordinates $\mathbf{q} = [u, v]$ via the projection equations:

$$\begin{aligned} f\frac{x'}{z'} &= u - u_0, \\ f\frac{y'}{z'} &= v - v_0, \end{aligned} \tag{3.1}$$

where u and v are pixel coordinates and $[u_0, v_0]$ is the principal point of the image plane. The image pixel coordinate system works in terms of pixels measured from the top left of the image and with a maximum of 5168×3448 pixels. We will assume that the principal point lies in the center of the image.

The focal length is described by f. For our camera system we have $f = 16$mm, so we have to convert the right-hand side of (3.1) to millimeters as well. Therefore we multiply with the ratio between the sensor dimensions and number of pixels, i.e. $s_u = \frac{36}{5168}$ and $s_v = \frac{24}{3448}$ for sensor dimension 36×24mm. For convenience, we only consider one scalar $s = s_u$ for $s_u \approx s_v$.

Now, the system (3.1) can be written in terms of millimeters as

$$\begin{aligned} f\frac{x'}{z'} &= (u-u_0)\,s, \\ f\frac{y'}{z'} &= (v-v_0)\,s. \end{aligned} \tag{3.2}$$

In addition, we can divide by f to arrive at the normalized sensor coordinates $\mathbf{y} = [f^{-1}\overline{u}, f^{-1}\overline{v}, 1]$ which are dimensionless, where $\overline{u} = (u-u_0)\,s$, $\overline{v} = (v-v_0)\,s$, i.e.,

$$\begin{aligned} \frac{x'}{z'} &= \frac{(u-u_0)\,s}{f}, \\ \frac{y'}{z'} &= \frac{(v-v_0)\,s}{f}, \end{aligned} \tag{3.3}$$

or

$$\frac{1}{z'}\mathbf{r}' = \begin{bmatrix} x'/z' \\ y'/z' \\ z'/z' \end{bmatrix} = \begin{bmatrix} \overline{u}/f \\ \overline{v}/f \\ f/f \end{bmatrix} = \mathbf{y}. \tag{3.4}$$

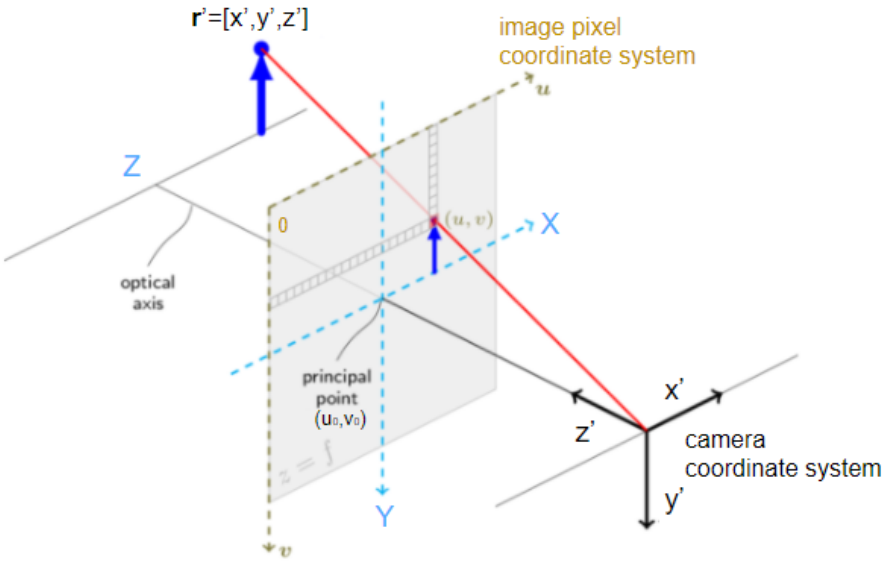


Figure 3.2: A pinhole camera model considering the different coordinate systems.

3.2. Essential matrix

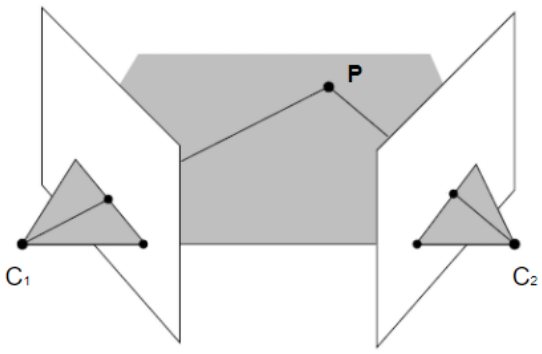


Figure 3.3: Scene with two cameras aimed at the same object point with the relative translation $\overrightarrow{C_2C_1}$ and a relative rotation described by the matrix R. Also shown the plane containing the origins of cameras' reference frames C_1 and C_2 and the object point P.

On the level of abstract vectors, we have the basic equality $\overrightarrow{C_2P} = \overrightarrow{C_2C_1} + \overrightarrow{C_1P}$ which holds in any reference frame and describes the relation between the object point P and the "origins" C_1 and C_2 of two cameras. Introducing reference frames associated with these two cameras, we identify $\mathbf{t} = [\overrightarrow{C_2C_1}]_2$ and $\mathbf{r}'' = [\overrightarrow{C_2P}]_2$ as the representations of the corresponding vectors in the reference frame of the second camera. At the same time $\mathbf{r}' = [\overrightarrow{C_1P}]_1$ is the representation of $\overrightarrow{C_1P}$ in the reference frame of the first camera. To get $[\overrightarrow{C_1P}]_2$ one only needs to apply the rotation R to $[\overrightarrow{C_1P}]_1$, i.e., $[\overrightarrow{C_1P}]_2 = R\mathbf{r}'$. Thus, the representation of the basic vector equality in the reference frame of the second camera, i.e., $[\overrightarrow{C_2P}]_2 = [\overrightarrow{C_2C_1}]_2 + [\overrightarrow{C_1P}]_2$, leads to the following transformation:

$$\mathbf{r}'' = \mathbf{t} + R\mathbf{r}', \tag{3.5}$$

where $\mathbf{r}' = \left[x', y', z'\right]^T$ are the coordinates of the object point P in the first camera's reference frame, $\mathbf{r}'' = \left[x'', y'', z''\right]^T$ are the coordinates of the same point P in the second camera's reference frame, and $\mathbf{t} = [t_1, t_2, t_3]$ contains the components of the translation vector, i.e., the coordinates of the origin C_1 of the first camera in the second camera's reference frame.
Since the three vectors $\overrightarrow{C_1P}$, $\overrightarrow{C_2P}$, and $\overrightarrow{C_2C_1}$ all lie in the same plane, they must satisfy the following *coplanarity* condition:

$$\overrightarrow{C_2P} \cdot (\overrightarrow{C_2C_1} \times \overrightarrow{C_1P}) = 0, \tag{3.6}$$

in any reference frame. A similar relation can also be derived algebraically from the transformation (3.5). For this purpose we associate with the translation vector $\mathbf{t}$ the following skew-symmetric matrix $[\mathbf{t}]_\times$:

$$[\mathbf{t}]_\times = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix},$$

with the properties:

$$\begin{aligned} \mathbf{t} \times \mathbf{a} &= [\mathbf{t}]_\times \mathbf{a}, \\ \mathbf{a}^T [\mathbf{t}]_\times \mathbf{a} &= 0, \\ [\mathbf{t}]_\times \mathbf{t} &= \mathbf{0}. \end{aligned}$$

We also note that using the pinhole projection equations of (3.4), we can rewrite $\mathbf{r}'$ in terms of the normalized sensor coordinates $\mathbf{y}'$ as

$$\mathbf{r}' = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = z'\mathbf{y}'. \tag{3.7}$$

Multiplying (3.5) with the matrix $[\mathbf{t}]_\times$, then with the transpose of $\mathbf{r}''$, and using (3.7), results in the *epipolar* (coplanarity) constraint for the so-called *essential matrix* E that involves only the normalized sensor coordinates $\mathbf{y}'$ and $\mathbf{y}''$ of any two matched points (i.e., images of the same object point):

$$\begin{aligned}
\mathbf{r}'' &= R\mathbf{r}' + \mathbf{t}, \\
[\mathbf{t}]_\times \mathbf{r}'' &= [\mathbf{t}]_\times R\mathbf{r}' + [\mathbf{t}]_\times \mathbf{t}, \\
[\mathbf{t}]_\times \mathbf{r}'' &= [\mathbf{t}]_\times R\mathbf{r}', \\
(\mathbf{r}'')^T [\mathbf{t}]_\times \mathbf{r}'' &= (\mathbf{r}'')^T [\mathbf{t}]_\times R\mathbf{r}', \\
0 &= (\mathbf{r}'')^T [\mathbf{t}]_\times R\mathbf{r}', \\
0 &= z'z''(\mathbf{y}'')^T [\mathbf{t}]_\times R\mathbf{y}', \quad z'z'' \neq 0, \\
0 &= (\mathbf{y}'')^T E\mathbf{y}',
\end{aligned} \tag{3.8}$$

where

$$E = [\mathbf{t}]_\times R, \tag{3.9}$$

is the decomposition of the essential matrix E in terms of the translation $\mathbf{t}$ and the rotation R.

Finding essential matrix from point correspondences

Having found a sufficient number of points correspondences, we can determine the essential matrix E. We rewrite the constraint (3.8) as:

$$\begin{aligned}
0 &= (\mathbf{y}'')^T E\mathbf{y}' = \mathrm{vec}((\mathbf{y}'')^T E\mathbf{y}') \\
&= (\mathbf{y}' \otimes \mathbf{y}'')^T \mathrm{vec}(E) = (\mathbf{y}' \otimes \mathbf{y}'')^T \mathbf{e}.
\end{aligned} \tag{3.10}$$

For n corresponding points we can further write this as

$$Y\mathbf{e} = \mathbf{0}, \tag{3.11}$$

where

$$Y = \begin{bmatrix}
f^{-2}\overline{u}_1''\overline{u}_1' & f^{-2}\overline{u}_1''\overline{v}_1' & f^{-2}v_1''\overline{u}_1' & f^{-2}\overline{v}_1''\overline{v}_1' & f^{-1}\overline{u}_1'' & f^{-1}\overline{v}_1'' & f^{-1}\overline{u}_1' & f^{-1}\overline{v}_1' & 1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\
f^{-2}\overline{u}_n''\overline{u}_n' & f^{-2}\overline{u}_n''\overline{v}_n' & f^{-2}\overline{v}_n''\overline{u}_n' & f^{-2}\overline{v}_n''\overline{v}_n' & f^{-1}\overline{u}_n'' & f^{-1}\overline{v}_n'' & f^{-1}\overline{u}_n' & f^{-1}\overline{v}_n' & 1
\end{bmatrix}.$$

We seek a value of $\mathbf{e}$ that minimizes $\|Y\mathbf{e}\|$ subject to the constraint $\|\mathbf{e}\| = \mathbf{e}^T\mathbf{e} = 1$, such that the trivial solution is avoided. A solution that minimizes $\|Y\mathbf{e}\|$ is the unit eigenvector corresponding to the smallest singular value of Y, or the smallest eigenvalue of Y^TY [4]. This solution can be found using the Singular Value Decomposition (SVD). This procedure of finding E is commonly known as the Eight-Point Algorithm, as it requires at least 8 points to solve the linear system up to scale [5].

Note that often for a singular value decomposition of matrices like Y a normalization is needed. This is indeed the case if pixel coordinates are used which often range from $(0, 5000)$. In our case the coordinates $\mathbf{y}$ are already normalized in the range of $(-\frac{18}{16}, \frac{18}{16})$ and $(-\frac{12}{16}, \frac{12}{16})$ if the points are evenly distributed over the image. When only a part of the image is considered, we can apply a transformation T such that the data is zero-centered between $(-1, 1)$, see [5],

$$\hat{\mathbf{y}} = T\mathbf{y}, \tag{3.12}$$

where

$$T = \begin{bmatrix}
\frac{1}{\sigma_x} & 0 & -\frac{\mu_x}{\sigma_x} \\
0 & \frac{1}{\sigma_y} & -\frac{\mu_y}{\sigma_y} \\
0 & 0 & 1
\end{bmatrix}. \tag{3.13}$$

The transformation of the data depends on the mean of the data μ_x, μ_y and the standard deviation σ_x, σ_y for both axes respectively. This can be substituted into (3.10) resulting in

$$0 = (\mathbf{y}'')^T E\mathbf{y}' = (T^{-1}\hat{\mathbf{y}}'')^T E T^{-1}\hat{\mathbf{y}}' = (\hat{\mathbf{y}}'')^T T^{-T} E T^{-1}\hat{\mathbf{y}}' = (\hat{\mathbf{y}}'')^T \hat{E}\hat{\mathbf{y}}', \tag{3.14}$$

which can be solved for $\hat{E}$ that relates to the original essential matrix E as

$$E = T^T \hat{E} T. \tag{3.15}$$

The essential matrix has a rank deficiency, but due to imperfect correspondence between the points, it is possible that the approximation of **e** doesn't meet this criterion. Also, for a 3×3 matrix to be an essential matrix, it has to have two identical singular values and the third must equal zero [5]. This can be shown by the definition $E = [\mathbf{t}]_\times R$, where the skew-symmetric matrix $[\mathbf{t}]_\times$ can be decomposed into $[\mathbf{t}]_\times = kUZU^T$, with U orthogonal, Z as below and a scalar k, see [6]. Indeed, let

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad Z = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{3.16}$$

Up to sign, $Z = \text{diag}(1,1,0)W$. Therefore,

$$\begin{aligned} E &= [\mathbf{t}]_\times R = kUZU^T R = kU\text{diag}(1,1,0)WU^T R, \\ &\text{with } W, U^T, R \text{ orthogonal matrices,} \\ E &= U\text{diag}(k,k,0)V^T. \end{aligned} \tag{3.17}$$

with $V^T = WU^T R$ an orthogonal matrix, i.e., we have a singular value decomposition with two identical singular values and one zero.

These properties can be enforced by yet another singular value decomposition, but this time applied on the rearranged vector **e**. Let $\mathbf{e}_{3\times3} = UDV^T$ be the singular value decomposition of $\mathbf{e}_{3\times3} = \text{mat}(\mathbf{e})$, where D is a diagonal matrix, $D = \text{diag}[r,s,t]$, satisfying $r \geq s \geq t$. The matrix deficiency constraint can be enforced by approximating E by $E' = U\text{diag}(r,s,0)V^T$. This method was suggested by Tsai and Huang [7] and minimizes $\|E - E'\|$, subject to the condition $\det E' = 0$. In order to enforce the two identical singular values, we can also approximate E by $E' = U\text{diag}(1,1,0)V^T$.

```
U, D, Vt = SVD(Y)
E = reshape(Vt[-1,:],(3,3))
U1, D1, V1t = SVD(E)
E = U1 * diag(1,1,0) * V1t
```

Listing 3.1: Algorithm for finding E

Scene reconstruction

The translation **t** can only be determined up to scale, so we will try to find the unit vector in the direction of **t**. To derive $\|\mathbf{t}\|$ we use that

$$EE^T = [\mathbf{t}]_\times RR^T [\mathbf{t}]_\times^T = -[\mathbf{t}]_\times^2 \tag{3.18}$$

and

$$\text{Tr}(EE^T) = \text{Tr}\begin{bmatrix} t_2^2 + t_3^2 & -t_1 t_2 & -t_1 t_3 \\ -t_1 t_2 & t_1^2 + t_3^2 & -t_2 t_3 \\ -t_1 t_3 & -t_2 t_3 & t_1^2 + t_2^2 \end{bmatrix} = 2\|\mathbf{t}\|^2, \tag{3.19}$$

such that $\|\mathbf{t}\| = \sqrt{\frac{\text{Tr}(EE^T)}{2}}$, see [8]. We can now find a normalized decomposition of the essential matrix.

To reconstruct R and **t** from the essential matrix E, we can use the result of (3.17) and the matrices of W and Z of (3.16), where $ZW = \text{diag}(1,1,0)$ as in [5]. Now,

$$E = U\text{diag}(1,1,0)V^T = UZWV^T = UZ\underbrace{U^T U}_{I}WV^T = \underbrace{UZU^T}_{[\mathbf{t}]_\times}\underbrace{UWV^T}_{R}. \tag{3.20}$$

However, this is not the only solution. There are four possibilities to define diag(1,1,0) in terms of W and Z:

$$\begin{bmatrix}1&0&0\\0&1&0\\0&0&0\end{bmatrix} = ZW = Z^T W^T = -Z^T W = -ZW^T.$$

So, we have to consider the two possibilities for the translation:

$$[\mathbf{t}]_\times = UZU^T \quad \text{or} \quad UZ^T U^T,$$

and the two possibilities for the rotation matrix:

$$R = UWV^T \quad \text{or} \quad UW^T V^T.$$

The only suitable solution is the solution where the points lie in front of both cameras. This has to be tested for the different combinations of R and $[\mathbf{t}]_\times$. This yields a unique, physically correct, configuration.

A 3D reconstruction using the rotation matrix R and translation vector $\mathbf{t}$ has been presented by Longuet-Higgins [9], however we will present a more general solution using a least-squares approximation. From (3.7) follows

$$\begin{aligned}
\mathbf{y}'' = \frac{1}{z''}\mathbf{r}'' &= \frac{R\mathbf{r}'+\mathbf{t}}{\mathbf{e}_3^T(R\mathbf{r}'+\mathbf{t})},\\
\mathbf{y}''\mathbf{e}_3^T(R\mathbf{r}'+\mathbf{t}) &= R\mathbf{r}'+\mathbf{t},\\
\mathbf{y}''\mathbf{e}_3^T(z'R\mathbf{y}'+\mathbf{t}) &= z'R\mathbf{y}'+\mathbf{t},\\
z'\mathbf{y}''\mathbf{e}_3^T R\mathbf{y}'+\mathbf{y}''\mathbf{e}_3^T\mathbf{t} &= z'R\mathbf{y}'+\mathbf{t},\\
z'\left(\mathbf{y}''\mathbf{e}_3^T R\mathbf{y}'-R\mathbf{y}'\right) &= \mathbf{t}-\mathbf{y}''\mathbf{e}_3^T\mathbf{t},\\
z'\left(\mathbf{y}''\mathbf{e}_3^T-I\right)R\mathbf{y}' &= -\left(\mathbf{y}''\mathbf{e}_3^T-I\right)\mathbf{t}.
\end{aligned} \tag{3.21}$$

The explicit least-squares solution for z' is given by:

$$z' = -\frac{\mathbf{y}'^T R^T\left(\mathbf{y}''\mathbf{e}_3^T-I\right)^T\left(\mathbf{y}''\mathbf{e}_3^T-I\right)\mathbf{t}}{\mathbf{y}'^T R^T\left(\mathbf{y}''\mathbf{e}_3^T-I\right)^T\left(\mathbf{y}''\mathbf{e}_3^T-I\right)R\mathbf{y}'}, \tag{3.22}$$

or

$$z' = -\frac{\mathbf{y}'^T R^T Y''\mathbf{t}}{\mathbf{y}'^T R^T Y'' R\mathbf{y}'}, \tag{3.23}$$

$$Y'' = \left(\mathbf{y}''\mathbf{e}_3^T-I\right)^T\left(\mathbf{y}''\mathbf{e}_3^T-I\right) = \begin{bmatrix}1&0&-f^{-1}\overline{u}''\\0&1&-f^{-1}\overline{v}''\\-f^{-1}\overline{u}''&-f^{-1}\overline{v}''&(f^{-1}\overline{u}'')^2+(f^{-1}\overline{v}'')^2\end{bmatrix}. \tag{3.24}$$

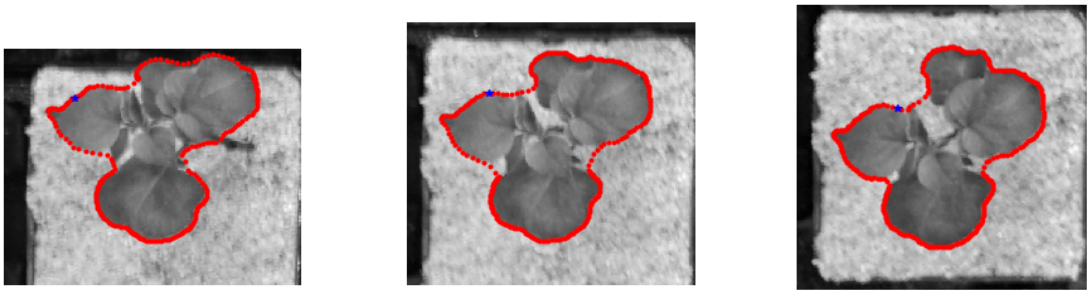
Note that $Y''\mathbf{y}'' = \mathbf{0}$, and the formula (3.23) becomes singular if $R\mathbf{y}' = \mathbf{y}''$. Thus, z' can not be determined unless there is a nontrivial rotation $R \neq I$ and/or translation $\mathbf{t} \neq \mathbf{0}$ between the two cameras.

	General Notation
$P = [x,y,z]^T$	Object point in world coordinates
$\mathbf{q} = [u,v]$	Point in pixel coordinates
$\mathbf{r}' = [x',y',z']^T$	Object point in first camera reference frame
$\mathbf{r}'' = [x'',y'',z'']^T$	Object point in second camera referece frame
$\mathbf{y}' = [f^{-1}\overline{u}', f^{-1}\overline{v}', 1]^T$	Normalized sensor coordinates of object in first camera ref. frame
$\mathbf{y}'' = [f^{-1}\overline{u}'', f^{-1}\overline{v}'', 1]^T$	Normalized sensor coordinates of object in second camera ref. frame

Table 3.1: General notation used in Chapter 3.

3.3. Point matching

To determine the essential matrix, point correspondences are needed. As we can see in Figure 3.4, points of the contours $\mathbf{q}(s_i)$ are not perfectly aligned with one another. We can clearly see $\mathbf{q}_1(0) \neq \mathbf{q}_2(0) \neq \mathbf{q}_3(0)$ and in general $s_1 \neq s_2 \neq s_3$. Therefore, the points have to be matched before we can continue with a 3D scene reconstruction.
Often methods as SIFT, SURF, or ORB in combination with RANSAC are used in computer vision, but they can be computationally expensive when used in large datasets. In the case of contour matching between a (mostly) shifted view, the problem is restrained to matching points only on the contour. This allows us to match point via a rough alignment and subsequently, apply RANSAC.



(a) Image and contour 1 (b) Image and contour 2 (c) Image and contour 3

Figure 3.4: Different parametrizations of $\mathbf{q}(s)$ visible for the three contours. The starting point of the contour $\mathbf{q}(0)$ is shown in blue.

RANSAC

Random sample consensus (RANSAC) is an iterative method to fit a model subject to (often noisy) data based on a random subset. It makes a distinction between inliers and outliers, to ignore noise [10].
For example, when fitting a line through a set of data points, 2 points are needed to define the line. To this end, two random points are selected in every iteration. The distance between other points and this line can serve as a threshold to measure the fitting of the model. If a point satisfies this threshold, it is added to the set of inliers. If a sufficient number of inliers is found, the algorithm terminates and there exists a line that fits the data, disregarding the outliers.

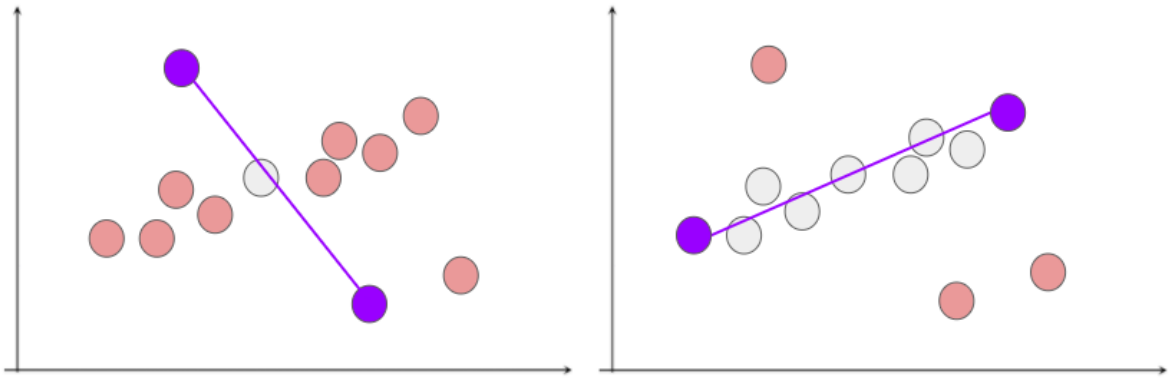


Figure 3.5: Example of RANSAC: fitting a line to datapoints. The algorithm iterates untill a sufficient number of inliers is found.

Rough alignment

In the current setup, a camera is moved along the table in one direction, resulting in a translation of the camera and the camera image. We can determine this translation in image coordinates as follows:
From (3.5) we have

$$\mathbf{r}'' = R\mathbf{r}' + \mathbf{t},$$

which can be transformed to the sensor coordinates $\mathbf{y}$ through (3.7) into

$$\mathbf{y}'' = \tilde{R}\mathbf{y}' + \tilde{\mathbf{t}}, \quad \text{where } \tilde{R} = \frac{z'R}{z''}, \quad \tilde{\mathbf{t}} = \frac{\mathbf{t}}{z''}. \tag{3.25}$$

The optimal translation can be found by minimizing the coordinate transform of (3.25) by taking the derivative with respect to $\tilde{\mathbf{t}}$ and setting it equal to zero, i.e.,

$$\begin{aligned} F(\tilde{\mathbf{t}}) &= \sum_{i=1}^{n} \|\tilde{R}\mathbf{y}'_i + \tilde{\mathbf{t}} - \mathbf{y}''_i\|_2^2 \\ \frac{\partial F}{\partial \tilde{\mathbf{t}}} &= 2\sum_{i=1}^{n} \left(\tilde{R}\mathbf{y}'_i + \tilde{\mathbf{t}} - \mathbf{y}''_i\right) = 0 \end{aligned} \tag{3.26}$$

So the optimal translation yields

$$\tilde{\mathbf{t}}_{\text{opt}} = \frac{\sum_{i=1}^{n} \mathbf{y}''_i - \sum_{i=1}^{n} \tilde{R}\mathbf{y}'_i}{n}, \tag{3.27}$$

and if, for a rough alignment, we disregard the rotation, i.e. $R = I$, the optimal translation is the distance between the centroids of the two curves. This is convenient because we don't have to take the starting point or parametrization of $\mathbf{y}(s)$ into account.
Note that this procedure is only possible for pure translations or transformations with only a small rotation.

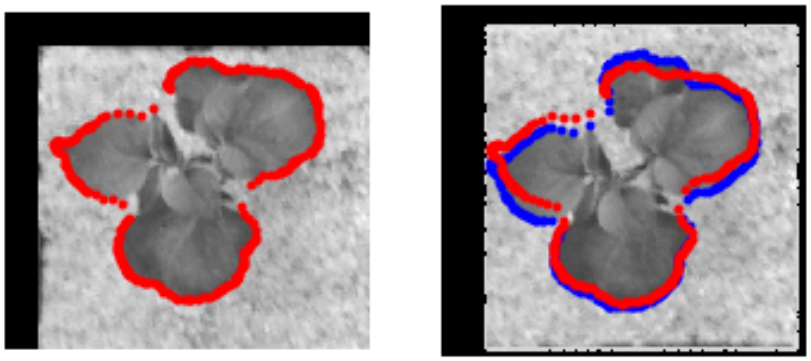


Figure 3.6: Result of rough alignment through translation. Transformation applied to contour 2 in image 3.

Now that the contours are more aligned, we can look for points closest nearby in terms of the Euclidean distance. As we can see in Figure 3.6, this approach is not applicable at all point of the contour, but, as we only need 8 points to estimate the essential matrix, we will choose 8 random points on the second contour and find the corresponding points in the transformed first contour via the minimal Euclidean distance. If these 8 matched points are well-chosen, the rest of the points will satisfy $\mathbf{y}''E\mathbf{y}'$ up to a certain value. If this is not the case, we will follow a RANSAC procedure and choose another set of 8 points until this threshold is met.

```
while it < k
    n2: find 8 random points on contour 2 with minimum intermediate distance l
    n1: find the points on transformed contour 1 closest to points n2 of contour 2

    Y = kron(y'(n1),y''(n2)).T
    U, D, Vt = SVD(Y)
    E = reshape(Vt[-1,:],(3,3))
    U1, D1, V1t = SVD(E)
    E = U1 * diag(1,1,0) * V1t

    for i=0:len(y'):
        if (y(i)''Ey(i)')<=eps:
```

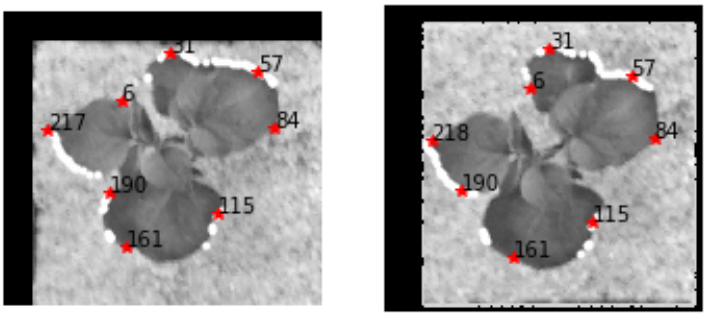
```
            inliers=[inliers,i]

    if len(inliers)>=d:
        Essential_matrix=E
    else:
        it=it+1
```

Listing 3.2: Algorithm for finding essential matrix with a RANSAC approach

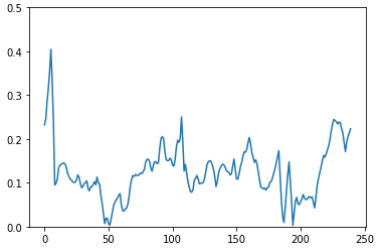
Numerical estimation of the essential matrix

For example, we can find an essential matrix where we want at least $d = 50$ points to satisfy $\mathbf{y}''E\mathbf{y}' \leq \varepsilon = 0.1$ Initially, we choose 8 points on the contour with an intermediate distance of $l = 25$ so that the points are distributed along the contour. We will run the algorithm for a maximum of $k = 4000$ iterations.

After the first iteration, 76 of the remaining points are added to the set of inliers for a threshold of $\varepsilon = 0.1$. The result is shown in Figure 3.7a, the 8 matched points are shown in red, and 6 of them are already matched quite well. The value of $\mathbf{y}''E\mathbf{y}'$ is shown per point in Figure 3.7b. The threshold of 0.1 is not always achieved.



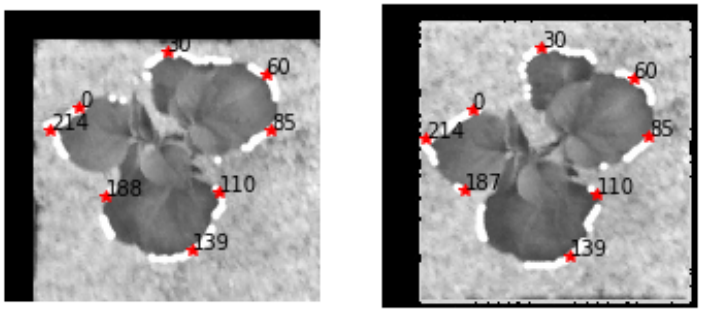
(a) Example of 8 matched points (red) and 67 inliers (white).



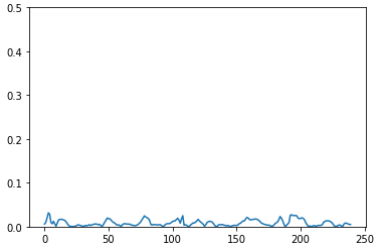
(b) $\mathbf{y}''E\mathbf{y}'$ per point of contour, with $\|\mathbf{y}''E\mathbf{y}'\|_2 = 2.16$.

Figure 3.7: Point matching based on rough transformation with a RANSAC threshold of 0.1 resulting in 67 inliers in the first iteration.

In order to minimize $\|\mathbf{y}''E\mathbf{y}'\|_2$ further, we can lower the threshold ε to 0.01. We seek a minimum of $d = 50$ inliers. This is achieved in 18 iterations and the matched points are shown in Figure 3.8a. Visually, we can see an improvement, except for point 188, which is still matched quite poorly. The values of $\mathbf{y}''E\mathbf{y}'$ shown in Figure 3.8b are very small and the norm $\|\mathbf{y}''E\mathbf{y}'\|_2$ is decreased to a value of 0.17.



(a) Example of 8 matched points (red) and 151 inliers (white).



(b) $\mathbf{y}''E\mathbf{y}'$ per point of contour, with $\|\mathbf{y}''E\mathbf{y}'\|_2 = 0.17$.

Figure 3.8: Point matching based on rough transformation with a RANSAC threshold of 0.01 resulting in 151 inliers after 18 iterations.

Equidistant points

As we can see in Figure 3.6, points on the contour are not evenly distributed and searching for the closest point

on the transformed contour in these parts of the contour can lead to unsatisfactory results. Therefore, we can make the points on the contour equidistant, such that the algorithm is more stable. Using this approach, searching for corresponding points performs best.

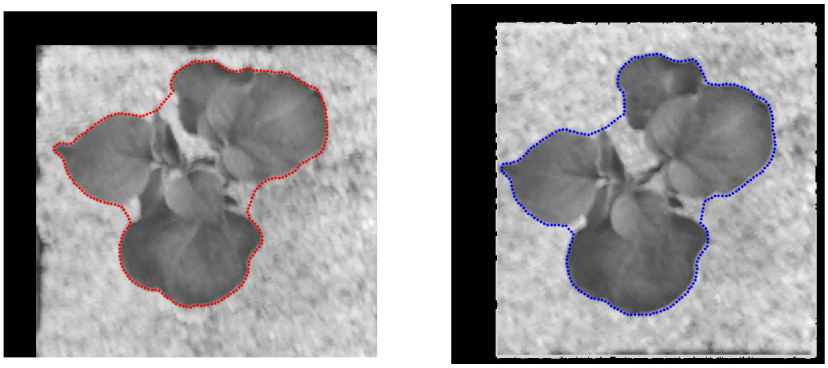


Figure 3.9: Contours with $n = 240$ equidistant points.

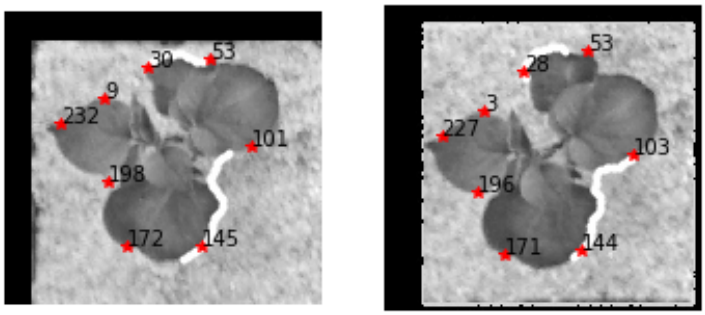


Figure 3.10: Example of 8 matched points (red) and sufficient number of inliers (white) based on rough transformation with a RANSAC threshold of 0.01 resulting in 75 inliers in the first iterations for equidistant points.

3.4. Reconstruction of the plant edge

The essential matrix $E = [\mathbf{t}]_\times R$ is only determined up to scale. Therefore, we want to include some extra points as reference points. In this Section we will make use of the (manually) matched corner points, such that we can reconstruct the container as well to measure the relative growth.

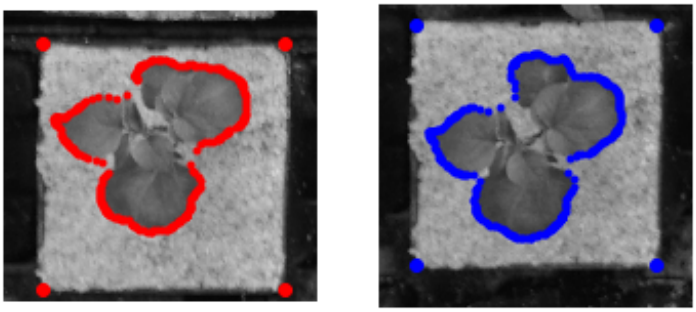
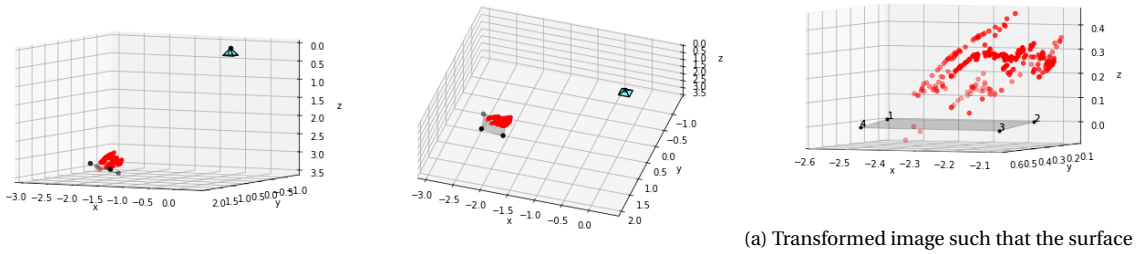


Figure 3.11: The contours for both views with (manually) chosen corners points. The corners are numbered clockwise starting at the top left.

The first reconstruction is based on every point on the contour, i.e., assuming that the images of all points are matched. There is only one configuration of $\mathbf{t}$ and R such that all points lie in front of the camera, that is,

$$\mathbf{t} = \begin{bmatrix} -0.28 \\ 0.87 \\ 0.40 \end{bmatrix}, \quad R = \begin{bmatrix} 0.998 & 0.026 & 0.051 \\ -0.030 & 0.996 & 0.089 \\ -0.049 & -0.090 & 0.995 \end{bmatrix}. \tag{3.28}$$

The reconstructed translation and rotation coincides with the movement of the camera over the table, which mainly moves along the y-axis. The plant should be located in the third quadrant as shown in Figure 3.13. This is indeed true for the reconstruction of the plant shown in 3.12. The container is approximately 0.4×0.5, where the maximum height of the plant is 0.46. However, when all points are considered, the results are strongly influenced by noise.



(a) Transformed image such that the surface of the container is plotted at $z = 0$.

Figure 3.12: Reconstruction of plant and container in the camera reference frame based on all points. The center of the camera is located at the origin.

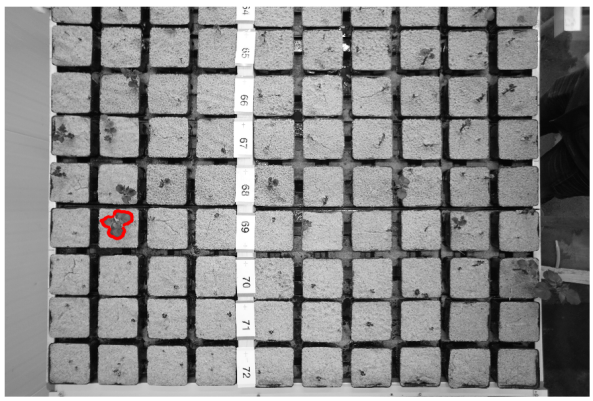
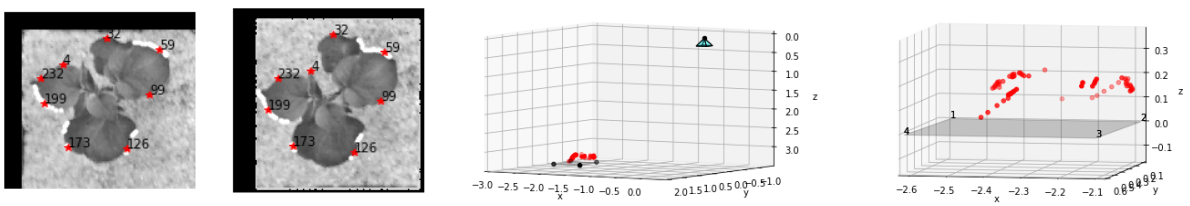


Figure 3.13: The actual set up of the first camera with the location of the plant that is attempted to reconstruct.

In the next reconstruction, we will make use of 8 points that are matched using the RANSAC-based algorithm.



(a) 8 matched points(red) and inliers(white).

(b) Scene reconstruction with camera at origin.

(c) Transformed image with container at $z = 0$.

Figure 3.14: Reconstruction of plant and container in the camera reference frame based on 8 points. The inliers are considered as matched points as well.

The combination of $\mathbf{t}$ and R such that the points lie in front of the camera yields,

$$\mathbf{t} = \begin{bmatrix} -6.45\text{e}{-14} \\ 1 \\ 9.09\text{e}{-14} \end{bmatrix}, \quad R = \begin{bmatrix} 1 & 6.66\text{e}{-15} & 8.10\text{e}{-15} \\ -6.59\text{e}{-15} & 1 & 8.13\text{e}{-16} \\ -8.16\text{e}{-15} & -7.44\text{e}{-16} & 1 \end{bmatrix}, \tag{3.29}$$

which equals, up to machine precision, a pure translation. The reconstructed height with respect to the container for the 8 points is also shown in Figure 3.15. We do see a resemblance of height, where point 32 is located on a leaf that is overgrown by another leaf that contains point 59.

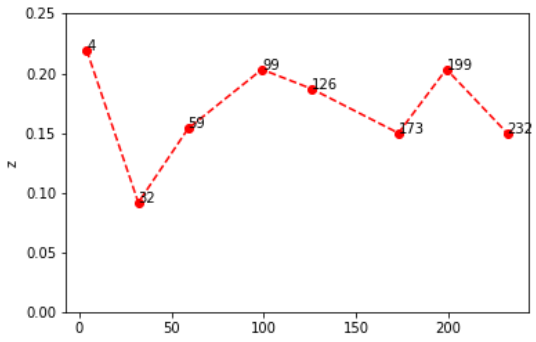
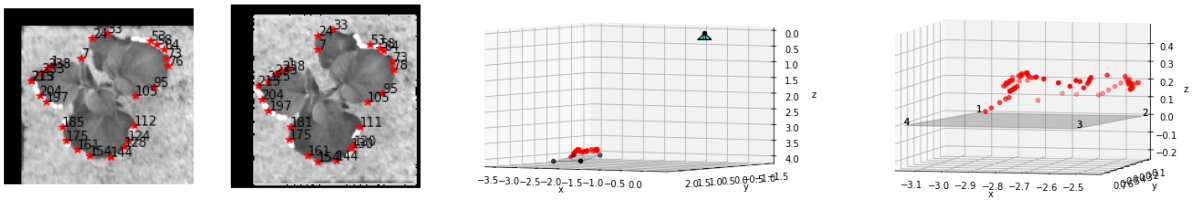


Figure 3.15: Height shown for the 8 points where E is based upon.

In general, the camera movement is not a pure translation, so we will find another configuration of the essential matrix based on 25 points matched with the RANSAC-based algorithm. The decomposition of E yields

$$\mathbf{t} = \begin{bmatrix} 0.08 \\ 0.99 \\ -0.11 \end{bmatrix}, \quad R = \begin{bmatrix} 1 & 0.015 & -0.010 \\ -0.014 & 1 & 0.025 \\ 0.011 & -0.025 & 1 \end{bmatrix}. \tag{3.30}$$



(a) 25 matched points(red) and inliers(white)

(b) Scene reconstruction with camera at origin.

(c) Transformed image with container at $z = 0$.

Figure 3.16: Reconstruction of plant and container in the camera reference frame based on 25 points. The inliers are considered as matched points as well

The container is about 0.6×0.6 and the maximum height of the plant with respect to the container is 0.26. For a container of 20×20 cm, this corresponds with a height of 8.7 cm. The most interesting result is that we can recognize the leaf 3D contour, where the tip of the leaf tends to go down, for point 64 and for point 213 in Figure 3.17.

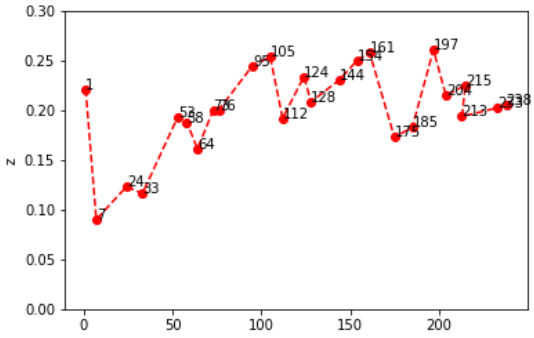


Figure 3.17: Height shown for the 25 points where E is based upon.

All of the previous results have been obtained by an extra normalization using (3.12) to obtain a stable singular value decomposition. Unfortunately, this doesn't mean that for every combination of carefully matched points and matrix E this results in a decomposition of $\mathbf{t}$ and R as we would expect. The essential matrix based on the 8 points of Figure 3.8a yields a decomposition of

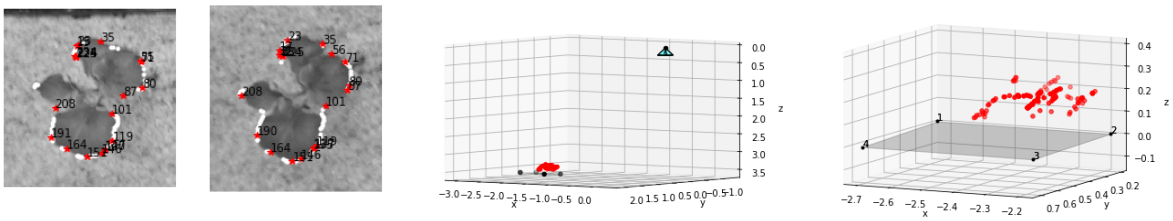
$$\mathbf{t} = \pm \begin{bmatrix} 0.62 \\ 0.05 \\ -0.78 \end{bmatrix}, \quad R_1 = \begin{bmatrix} 0.50 & -0.85 & 0.15 \\ 0.72 & 0.51 & 0.48 \\ -0.48 & -0.13 & 0.86 \end{bmatrix}, \quad R_2 = \begin{bmatrix} 0.40 & 0.35 & -0.85 \\ -0.65 & -0.54 & -0.53 \\ -0.64 & 0.76 & 0.01 \end{bmatrix}, \tag{3.31}$$

which does not result in a physically correct configuration for the camera's, i.e. it is not able to reconstruct the points in front of the camera. This may be caused by a poor choice of points or by the decomposition of the essential matrix, which is very sensitive. Another approach of recovering the translation and rotation, which is based on the cofactors of E and avoids a normalization and singular value decomposition, can then be investigated[8][11].

3.5. Observation of growth over time

To reconstruct the relative growth of a plant, we can make a 3D reconstruction on a preceding time step. In Figure 3.19 the reconstruction is shown based on images taken 3 days before the images in Section 3.4.

Figure 3.18: Two images of a plant with an interval of 3 days.



(a) 20 matched points(red) and inliers(white)

(b) Scene reconstruction with camera at origin

(c) Transformed image with container at $z = 0$.

Figure 3.19: Reconstruction of plant and container in the camera reference frame on a previous timestep.

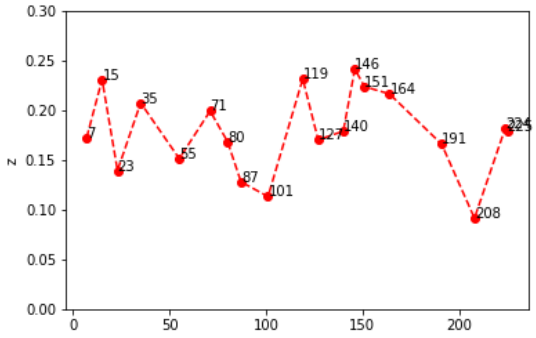


Figure 3.20: Height shown for reconstruction on previous timestep.

In order to compare this with the reconstruction in the previous Section, another transformation between the camera reference frames on times τ_1 and τ_2 has to be determined.

$$\mathbf{r}'(\tau_1) = aR\mathbf{r}'(\tau_2) + \mathbf{t} \tag{3.32}$$

In general, 4 points are needed to determine such a transformation. In the current setup, we do not have information about corresponding points in these consecutive snapshots, except for the points on the container. Unfortunately, these points are co-planar and linearly dependent such that we can not solve (3.32) in a general way.

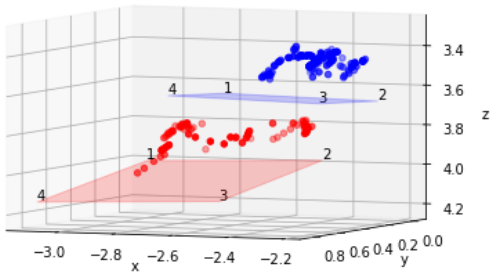


Figure 3.21: The two reconstructions for the plant and container at τ_1 in blue and τ_2 in red.

Pot alignment

Let $\mathbf{r}'_n(\tau_2)$, $n = 1,2,\ldots$ be all reconstructed points on the date τ_2. For comparison purposes, we want to transform them to same scale and reference frame as the one of the points $\mathbf{r}'_m(\tau_1)$, $m = 1,2,\ldots$ reconstructed on the date τ_1. To achieve this we use $\mathbf{r}'_i(\tau)$, $i = 1,2,3,4$; the same four pot corner points reconstructed on both dates. First, we express all reconstructed points on both dates via their relative coordinates with respect to one of the pot corners, e.g., point $\mathbf{r}'_1$. The transformation between these relative reference frames is:

$$\tilde{\mathbf{r}}'(\tau_2) - \mathbf{r}'_1(\tau_1) = aR[\mathbf{r}'(\tau_2) - \mathbf{r}'_1(\tau_2)]. \tag{3.33}$$

Hence, the reconstructed points $\mathbf{r}'(\tau_2)$ on the date τ_2 can be transformed to the scale and frame associated with the date τ_1, i.e., into the new points $\tilde{\mathbf{r}}'(\tau_2)$, as follows:

$$\tilde{\mathbf{r}}'(\tau_2) = \mathbf{r}'_1(\tau_1) + aR[\mathbf{r}'(\tau_2) - \mathbf{r}'_1(\tau_2)]. \tag{3.34}$$

To recover the scaling and rotation we use the remaining three corner points. For these time-invariant points it should hold:

$$\tilde{\mathbf{r}}'_i(\tau_2) = \mathbf{r}'_i(\tau_1), \quad i = 2,3,4. \tag{3.35}$$

Hence, the scaling and rotation can be recovered by solving the linear system:

$$\begin{bmatrix} \mathbf{r}'_2(\tau_1) - \mathbf{r}'_1(\tau_1) \\ \mathbf{r}'_3(\tau_1) - \mathbf{r}'_1(\tau_1) \\ \mathbf{r}'_4(\tau_1) - \mathbf{r}'_1(\tau_1) \end{bmatrix} = a \begin{bmatrix} [\mathbf{r}'_2(\tau_2) - \mathbf{r}'_1(\tau_2)]^T & \mathbf{0}^T & \mathbf{0}^T \\ \mathbf{0}^T & [\mathbf{r}'_2(\tau_2) - \mathbf{r}'_1(\tau_2)]^T & \mathbf{0}^T \\ \mathbf{0}^T & \mathbf{0}^T & [\mathbf{r}'_2(\tau_2) - \mathbf{r}'_1(\tau_2)]^T \\ [\mathbf{r}'_3(\tau_2) - \mathbf{r}'_1(\tau_2)]^T & \mathbf{0}^T & \mathbf{0}^T \\ \mathbf{0}^T & [\mathbf{r}'_3(\tau_2) - \mathbf{r}'_1(\tau_2)]^T & \mathbf{0}^T \\ \mathbf{0}^T & \mathbf{0}^T & [\mathbf{r}'_3(\tau_2) - \mathbf{r}'_1(\tau_2)]^T \\ [\mathbf{r}'_4(\tau_2) - \mathbf{r}'_1(\tau_2)]^T & \mathbf{0}^T & \mathbf{0}^T \\ \mathbf{0}^T & [\mathbf{r}'_4(\tau_2) - \mathbf{r}'_1(\tau_2)]^T & \mathbf{0}^T \\ \mathbf{0}^T & \mathbf{0}^T & [\mathbf{r}'_4(\tau_2) - \mathbf{r}'_1(\tau_2)]^T \end{bmatrix} \begin{bmatrix} r_{11} \\ r_{12} \\ r_{13} \\ r_{21} \\ r_{22} \\ r_{23} \\ r_{31} \\ r_{32} \\ r_{33} \end{bmatrix}, \tag{3.36}$$

$$\mathbf{d}_1 = aD_2\mathbf{x}$$

The system matrix D_2 is singular if the three vectors $\mathbf{r}'_i(\tau_2) - \mathbf{r}'_1(\tau_2)$, $i = 2,3,4$; are co-planar, which is almost certainly the case with the pot corner points. If the points are not lying on a line, but the vectors happen to be co-planar, the rank of the system matrix equals six. In this case, the minimum-norm solution of (3.36) could be used as an estimate of aR. The resulting rotation could be mirrored and should be corrected based on the proper location of the remaining reconstructed points with respect to the pot plane.

A better reconstruction of R can be achieved by enforcing the orthogonality property $R^TR = I$. To this end, we consider the additional linear system involving R^T:

$$a \begin{bmatrix} \mathbf{r}'_2(\tau_2) - \mathbf{r}'_1(\tau_2) \\ \mathbf{r}'_3(\tau_2) - \mathbf{r}'_1(\tau_2) \\ \mathbf{r}'_4(\tau_2) - \mathbf{r}'_1(\tau_2) \end{bmatrix} = \begin{bmatrix} [\mathbf{r}'_2(\tau_1) - \mathbf{r}'_1(\tau_1)]^T & \mathbf{0}^T & \mathbf{0}^T \\ \mathbf{0}^T & [\mathbf{r}'_2(\tau_1) - \mathbf{r}'_1(\tau_1)]^T & \mathbf{0}^T \\ \mathbf{0}^T & \mathbf{0}^T & [\mathbf{r}'_2(\tau_1) - \mathbf{r}'_1(\tau_1)]^T \\ [\mathbf{r}'_3(\tau_1) - \mathbf{r}'_1(\tau_1)]^T & \mathbf{0}^T & \mathbf{0}^T \\ \mathbf{0}^T & [\mathbf{r}'_3(\tau_1) - \mathbf{r}'_1(\tau_1)]^T & \mathbf{0}^T \\ \mathbf{0}^T & \mathbf{0}^T & [\mathbf{r}'_3(\tau_1) - \mathbf{r}'_1(\tau_1)]^T \\ [\mathbf{r}'_4(\tau_1) - \mathbf{r}'_1(\tau_1)]^T & \mathbf{0}^T & \mathbf{0}^T \\ \mathbf{0}^T & [\mathbf{r}'_4(\tau_1) - \mathbf{r}'_1(\tau_1)]^T & \mathbf{0}^T \\ \mathbf{0}^T & \mathbf{0}^T & [\mathbf{r}'_4(\tau_1) - \mathbf{r}'_1(\tau_1)]^T \end{bmatrix} \begin{bmatrix} r_{11} \\ r_{21} \\ r_{31} \\ r_{12} \\ r_{22} \\ r_{32} \\ r_{13} \\ r_{23} \\ r_{33} \end{bmatrix}, \tag{3.37}$$

$$a\mathbf{d}_2 = D_1P\mathbf{x},$$

where P is the permutation matrix:

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.38}$$

Combining (3.36) and (3.37) we get the normal equation:

$$\left[a^2 D_2^T D_2 + P^T D_1^T D_1 P\right]\mathbf{x} = a D_2^T \mathbf{d}_1 + a P^T D_1^T \mathbf{d}_2. \tag{3.39}$$

This problem is nonlinear, since both $\mathbf{x}$ and a are unknown. We will assume that the system matrix $a^2 D_2^T D_2 + P^T D_1^T D_1 P$ is full rank, as the null-spaces of D_2 and $D_1 P$ will not overlap, in general. One way to solve (3.39) is to rewrite it as the minimization problem for the function:

$$f(a,\mathbf{x}) = \left\| a D_2^T \mathbf{d}_1 + a P^T D_1^T \mathbf{d}_2 - \left[a^2 D_2^T D_2 + P^T D_1^T D_1 P\right]\mathbf{x} \right\|_2^2, \tag{3.40}$$

and to apply a nonlinear least-squares solver to it. Alternatively, one could attempt to solve (3.39) with an alternating directions method. Specifically, assuming that the matrix is full rank, we may proceed as follows:

$$\begin{aligned}
&a_0 = 1, \\
&\mathbf{x}_0 = \left[a_0^2 D_2^T D_2 + P^T D_1^T D_1 P\right]^{-1} (a_0 D_2^T \mathbf{d}_1 + a_0 P^T D_1^T \mathbf{d}_2), \\
&a_1 = -\frac{b_0}{2} \pm \sqrt{\left(\frac{b_0}{2}\right)^2 - c_0}, \quad \text{where} \quad \begin{cases} b_0 &= -\frac{\mathbf{x}_0^T D_2^T D_2 (D_2^T \mathbf{d}_1 + P^T D_1 \mathbf{d}_1)}{\|D_2^T D_2 \mathbf{x}_0\|_2^2}, \\ c_0 &= -\frac{\mathbf{x}_0^T D_2^T D_2 P^T D_1^T D_1 P \mathbf{x}_0}{\|D_2^T D_2 \mathbf{x}_0,\|_2^2} \end{cases} \\
&\mathbf{x}_1 = \left[a_1^2 D_2^T D_2 + P^T D_1^T D_1 P\right]^{-1} (a_1 D_2^T \mathbf{d}_1 + a_1 P^T D_1^T \mathbf{d}_2), \\
&a_2 = -\frac{b_1}{2} \pm \sqrt{\left(\frac{b_1}{2}\right)^2 - c_1}, \quad \text{where} \quad \begin{cases} b_1 &= -\frac{\mathbf{x}_1^T D_2^T D_2 (D_2^T \mathbf{d}_1 + P^T D_1 \mathbf{d}_1)}{\|D_2^T D_2 \mathbf{x}_1\|_2^2}, \\ c_1 &= -\frac{\mathbf{x}_1^T D_2^T D_2 P^T D_1^T D_1 P \mathbf{x}_1}{\|D_2^T D_2 \mathbf{x}_1,\|_2^2} \end{cases} \\
&\quad \text{etc.}
\end{aligned} \tag{3.41}$$

each time choosing the appropriate sign and magnitude out of the two possible solutions for the scaling factor a_k.
An alternative, unique, way to update the scaling factor a after updating the vector $\mathbf{x}$ is to compute it directly from (3.36) as follows:

$$a_{k+1} = \frac{\mathbf{x}_k^T D_2^T \mathbf{d}_1}{\|D_2 \mathbf{x}_k\|_2^2}. \tag{3.42}$$

Using the approach of (3.42) the scheme converges in 3 iterations and scaling of $a = 0.8899$ is obtained, which seems reasonable from the original images shown in Figure 3.21. The rotation matrix obtained from $\mathbf{x}_{3\times3}$ initially has determinant $\det(R) = -0.9612$. The correct rotation matrix can be found by changing the sign of the third column of R.

The result of this alignment is shown in Figure 3.22. The first corner is exactly aligned due to (3.34). The change in the alignment of all the corners are respectively,

$$\mathbf{r}_i(\tau_1) - \tilde{\mathbf{r}}_i(\tau_2) = \begin{bmatrix} 0 & 0.0006 & 0.0335 & 0.0613 \\ 0 & 0.0666 & 0.0300 & -0.0462 \\ 0 & 0.0388 & -0.0487 & -0.0909 \end{bmatrix}, i = 1, \ldots, 4. \tag{3.43}$$

With this alignment, we can make a comparison between the consecutive snapshots.

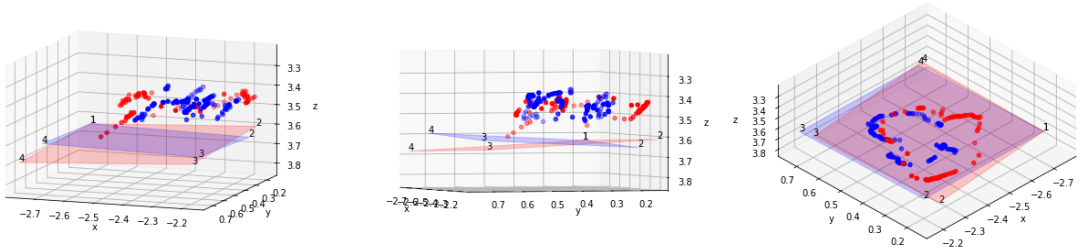


Figure 3.22: 3D reconstruction of plant at τ_1 in blue and τ_2 in red after pot alignment.

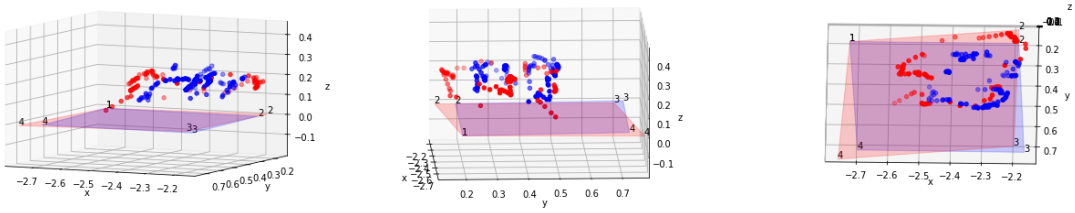


Figure 3.23: 3D reconstruction of plant at τ_1 in blue and τ_2 in red after pot alignment transformed such that surface of container is at $z = 0$.

As we can see in Figure 3.23, the outer edge of the plant evolves in size rather than in height in the considered time interval. Especially the leaf on the left side (side of corner 1 and corner 4), has evolved in size where the tip of the leaf points a bit downwards.

With the estimation and decomposition of the essential matrix applied to the two two-dimensional contours of the consecutive snapshots, we can conclude that it is indeed possible to reconstruct the three-dimensional growth of the plant in a certain time interval, up to certain errors made in the two reconstructions, compounded by the errors made in the alignment of the containers. To improve the approach considered in this Chapter, one can try to stabilize the RANSAC-based approach of Section 3.3 by imposing additional constraints, e.g., on the essential matrix.

Another room for improvement lies in the pot alignment algorithm of this Section, as it is only correct in the least-squares sense and is based on the simple model (3.32). The results of pot alignment, shown in Figures 3.22, 3.23, indicates that the differences between contour reconstructions at two dates do not fully satisfy the expected mathematical model (3.32) and involve anisotropic scaling in space. This is the result of imperfect reconstructions of the individual contours. It is reasonable to assume that the major source of errors is the essential matrix, since all other steps are mathematically trivial and numerically stable. To improve the estimate of the essential matrix, one needs to improve the point matching procedure, which is a known difficult problem in computer vision. Also, a more advanced camera model may be considered and a bundle-adjustment algorithm can be applied.

4

Double Snakes

In this Chapter a new method is proposed that combines the problems of contour finding and point matching into a single problem. The idea is to evolve two "snakes" in two images simultaneously, so that not only the plants' edge is found in both images, but also the points of the two resulting contours are matched. The result can then be directly employed in the three-dimensional reconstruction algorithm of the previous Chapter.

4.1. Variational formulation with coplanarity constraint

As we have seen, the contours $\mathbf{q}'(s)$ and $\mathbf{q}''(s)$ found by separately solving the GVF-Snake equations (2.12) on two images will correspond to the same three-dimensional closed curve. However, in general, the parametrizations of these contours will be such that the points $\mathbf{q}'(s)$ and $\mathbf{q}''(s)$, corresponding to the same value of the parameter s, will not be the images of the same point on that curve. We have also seen that it is possible to reconstruct the essential matrix E, since, with enough points, it is always possible to find at least some of them that satisfy the constraint (3.8).
To combine the epipolar (co-planarity) constraint (3.8), involving the essential matrix E, and the GVF-Snake equation (2.12), that is used to find the contours, we consider the following functional for the two contours $\mathbf{q}_1(s)$ and $\mathbf{q}_2(s)$ in terms of their normalized image coordinates $\mathbf{y}_1(s)$ and $\mathbf{y}_2(s)$:

$$\Phi[\mathbf{y}_1,\mathbf{y}_2] = \int_0^1 \left[\frac{\alpha}{2}\left(\|\mathbf{y}_1'\|^2 + \|\mathbf{y}_2'\|^2\right) + \frac{\beta}{2}\left(\|\mathbf{y}_1''\|^2 + \|\mathbf{y}_2''\|^2\right) + \frac{1}{2}\left(\mathbf{y}_2^T E \mathbf{y}_1\right)^2 \right] \mathrm{d}s \tag{4.1}$$

Here, for simplicity, we assume that the parameters α and β controlling the regularity are the same for both contours. Considering one of the contours, say, $\mathbf{y}_1(s)$, as well as the essential matrix E to be given or fixed, the Euler-Lagrange equation for $\mathbf{y}_2(s)$ is:

$$-\alpha\mathbf{y}_2'' + \beta\mathbf{y}_2'''' + E\mathbf{y}_1\mathbf{y}_1^T E^T\mathbf{y}_2 = \mathbf{0}. \tag{4.2}$$

Introducing the time variable in $\mathbf{y}_2(s,t)$, and the driving force $\mathbf{F}(\mathbf{y}_2)$ as in (2.12), we arrive at

$$\frac{\partial \mathbf{y}_2}{\partial t} = \alpha\frac{\partial^2\mathbf{y}_2}{\partial s^2} - \beta\frac{\partial^4\mathbf{y}_2}{\partial s^4} - E\mathbf{y}_1\mathbf{y}_1^T E^T\mathbf{y}_2 + \mathbf{F}(\mathbf{y}_2). \tag{4.3}$$

Since, by definition, the last component of $\mathbf{y}_2$ equals 1, we only need to find its first two components. To this end we introduce the vector $\boldsymbol{\gamma} = \left[f^{-1}\overline{u}, f^{-1}\overline{v}\right]^T$, the matrix Q, which is the first 2×2 block of the matrix $E\mathbf{y}_1\mathbf{y}_1^T E^T$, and the vector $\mathbf{w}_0$ consisting of the two upper elements of the last (third) column of $E\mathbf{y}_1\mathbf{y}_1^T E^T$. The corresponding equation for $\boldsymbol{\gamma}$ is:

$$\frac{\partial \boldsymbol{\gamma}}{\partial t} = \alpha\frac{\partial^2\boldsymbol{\gamma}}{\partial s^2} - \beta\frac{\partial^4\boldsymbol{\gamma}}{\partial s^4} - Q\boldsymbol{\gamma} - \mathbf{w}_0 + \mathbf{w}(\mathbf{q}). \tag{4.4}$$

Now, the relation between $\mathbf{q}$ and $\boldsymbol{\gamma}$ is as follows:

$$\boldsymbol{\gamma} = \begin{bmatrix} f^{-1}\overline{u} \\ f^{-1}\overline{v} \end{bmatrix} = f^{-1}\left(\begin{bmatrix} u \\ v \end{bmatrix} - \begin{bmatrix} u_0 \\ v_0 \end{bmatrix}\right) = f^{-1}\left(\mathbf{q} - \mathbf{c}\right),$$

with $\mathbf{c}$ being the principal point (optical center of the image). Using the chain rule, we arrive at the following equation in terms of the non-normalized and non-centered image coordinate vector $\mathbf{q}$:

$$f^{-1}\frac{\partial \mathbf{q}}{\partial t} = \alpha f^{-1}\frac{\partial^2 \mathbf{q}}{\partial s^2} - \beta f^{-1}\frac{\partial^4 \mathbf{q}}{\partial s^4} - Q\left(f^{-1}(\mathbf{q}-\mathbf{c})\right) - \mathbf{w}_0 + \mathbf{w}(\mathbf{q}), \quad \text{or} \tag{4.5}$$

$$\frac{\partial \mathbf{q}}{\partial t} = \alpha\frac{\partial^2 \mathbf{q}}{\partial s^2} - \beta\frac{\partial^4 \mathbf{q}}{\partial s^4} - Q\mathbf{q} + Q\mathbf{c} - f\mathbf{w}_0 + f\mathbf{w}(\mathbf{q}). \tag{4.6}$$

Let $\mathbf{q}(s)$ be the steady-state solution of this equation and suppose that $-Q\left((\mathbf{q}-\mathbf{c})\right) - f\mathbf{w}_0 = \mathbf{0}$. Then, for each s, the two points $\mathbf{y}_1(s)$ and $\mathbf{y}_2(s)$ are matched in the sense of the co-planarity condition (3.8), i.e., $\mathbf{y}_2^T(s)E\mathbf{y}_1(s) = 0$. If, on the other hand, $-Q\left((\mathbf{q}-\mathbf{c})\right) - f\mathbf{w}_0 \neq \mathbf{0}$, then a new essential matrix E_1 can be computed, such that $-Q_1\left((\mathbf{q}-\mathbf{c})\right) - f\mathbf{w}_1 = \mathbf{0}$ and the whole procedure repeated to find a new $\mathbf{q}$, etc.

4.2. Numerical experiments

To discretize (4.6), the extra terms $-Q\mathbf{q} + Q\mathbf{c} - f\mathbf{w}_0$ has to be discretized in addition to (2.17).

Remember that for every point $i = 1,\ldots,n$ on the contour we have

$$Q_i = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix}_i,$$

the first 2×2 block matrix of $E(\mathbf{y}_1)_i(\mathbf{y}_1)_i^T E^T$ and the vector

$$(\mathbf{w}_0)_i = \begin{bmatrix} Q_{13} \\ Q_{23} \end{bmatrix}_i,$$

consisting of the two upper elements of the last column of $E(\mathbf{y}_1)_i(\mathbf{y}_1)_i^T E^T$. We can vectorize these elements for every point i such that

$$\tilde{\mathbf{q}}_{jk} = \begin{bmatrix} (Q_{jk})_1, \\ \vdots \\ (Q_{jk})_n \end{bmatrix} \quad j = 1,2, \quad k = 1,2,3.$$

By introducing the vectors

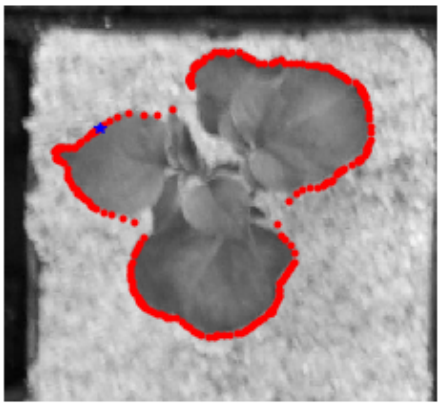
$$\begin{aligned} \mathbf{g}_u &= \tilde{\mathbf{q}}_{11}u_0 + \tilde{\mathbf{q}}_{12}v_0 - f\tilde{\mathbf{q}}_{13}, \\ \mathbf{g}_v &= \tilde{\mathbf{q}}_{21}u_0 + \tilde{\mathbf{q}}_{22}v_0 - f\tilde{\mathbf{q}}_{23} \end{aligned}$$

we can now discretize (4.6) as

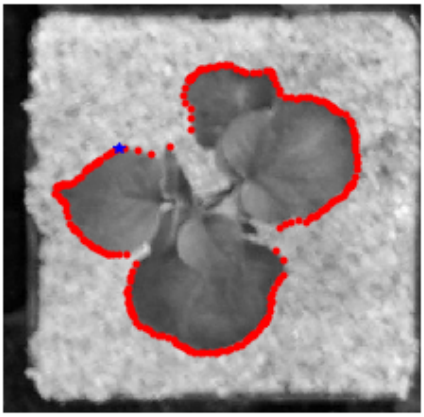
$$\begin{aligned} \mathbf{u}_{k+1} &= \mathbf{u}_k + \Delta t\left(\left(\alpha D_2 - \beta D_4 - \mathrm{diag}(\tilde{\mathbf{q}}_{11})\right)\mathbf{u}_k - \mathrm{diag}(\tilde{\mathbf{q}}_{12})\mathbf{v}_k + \mathbf{g}_u + f\gamma_u\boldsymbol{\phi}_k\right), \\ \mathbf{v}_{k+1} &= \mathbf{v}_k + \Delta t\left(\left(\alpha D_2 - \beta D_4 - \mathrm{diag}(\tilde{\mathbf{q}}_{22})\right)\mathbf{v}_k - \mathrm{diag}(\tilde{\mathbf{q}}_{21})\mathbf{u}_k + \mathbf{g}_v + f\gamma_v\boldsymbol{\psi}_k\right). \end{aligned} \tag{4.7}$$

To control the force of the extra terms we introduce an extra parameter λ such that

$$\begin{aligned} \mathbf{u}_{k+1} &= \mathbf{u}_k + \Delta t\left((\alpha D_2 - \beta D_4)\mathbf{u}_k + \lambda\left(-\mathrm{diag}(\tilde{\mathbf{q}}_{11})\mathbf{u}_k - \mathrm{diag}(\tilde{\mathbf{q}}_{12})\mathbf{v}_k + \mathbf{g}_u\right) + f\gamma_u\boldsymbol{\phi}_k\right), \\ \mathbf{v}_{k+1} &= \mathbf{v}_k + \Delta t\left((\alpha D_2 - \beta D_4)\mathbf{v}_k + \lambda\left(-\lambda\mathrm{diag}(\tilde{\mathbf{q}}_{22})\mathbf{v}_k - \mathrm{diag}(\tilde{\mathbf{q}}_{21})\mathbf{u}_k + \mathbf{g}_v\right) + f\gamma_v\boldsymbol{\psi}_k\right). \end{aligned} \tag{4.8}$$



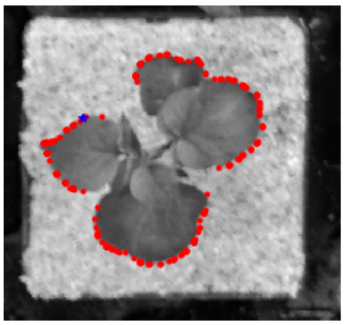
(a) Image and contour 1



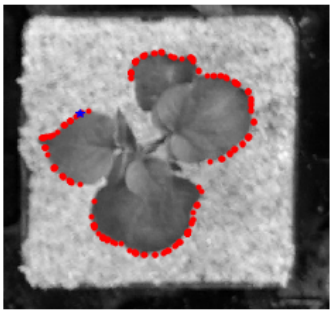
(b) Image and contour 2

Figure 4.1: Original GVF snake applied on images with contour of 240 points, $\Delta t = 0.01$. The starting point of the contour is shown in blue.

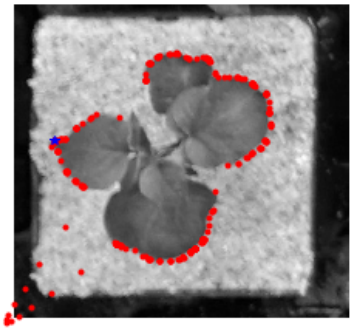
We will perform some numerical experiments with (4.8) for different values of λ. The initial contour is obtained by a square around the plant as in Chapter 2. The time-stepping is decreased due to stability reasons to $\Delta t = 0.001$. The rest of the parameters are kept the same. The essential matrix is obtained by the points shown in Figure 3.16, where the essential matrix can be deconstructed as $E = U\text{diag}(1,1,0)V^T$.



(a) $\lambda = 1e{-}4$



(b) $\lambda = 1e{-}3$

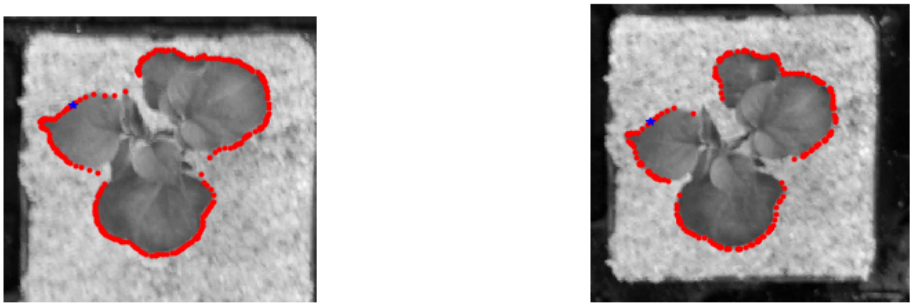


(c) $\lambda = 1e{-}2$

Figure 4.2: Double snake applied on images with contour of 240 points, $\Delta t = 0.001$, $t_{\text{end}} = 10$. The starting point is shown in blue.

As we can see in Figure 4.2, the final contour is detected for $\lambda = 1e{-}4$ and $\lambda = 1e{-}3$. This is an interesting result because the original GVF snakes algorithm takes $t = 40$ with the same settings. The contour is different from Figure 4.1, the points are accumulated and less smooth. We do not observe the expected shift in the starting point except when we set $\lambda = 1e{-}2$. However, then, the starting point is shifted too far along the contour in beyond the location that can be deduced by visually inspecting the starting point of the first contour.

We have also experimented with fine-tuning the parameter α hoping to obtain smoother results. In Figure 4.3 the result is shown for $\alpha = 5$ and $\lambda = 1e{-}2$. The points of the contour become less accumulated and the starting point of the contour is shifted to the left, as expected.

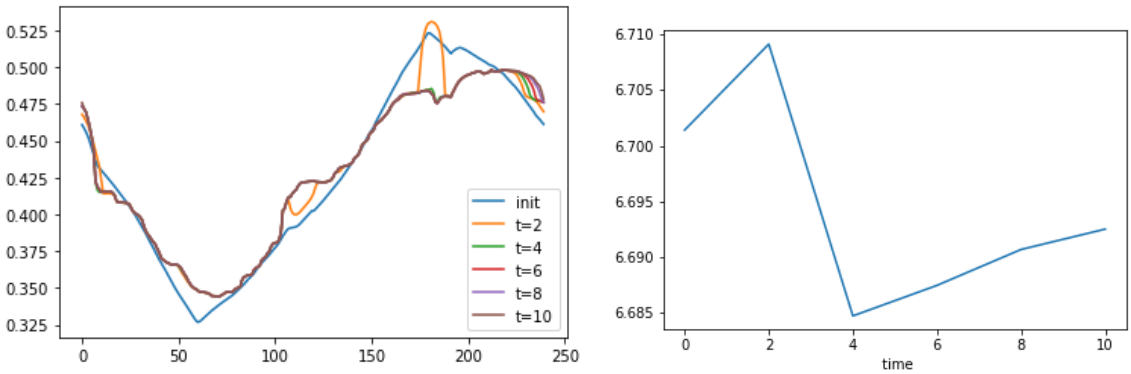


(a) Original contour 1

(b) Double snakes contour 2

Figure 4.3: Original Snakes contour and the Double Snakes contour (same image) for $\lambda = 1\text{e}{-2}, \alpha = 5, t_{\text{end}} = 10$.

The norm of the coplanarity term $\left\|\mathbf{y}_2^T E \mathbf{y}_1\right\|$ is only decreasing during the first time steps, indicating that we should not continue after $t_{\text{end}} = 4$. However, the norm of the coplanarity term obtained after applying the original decoupled Snakes appears to be lower than the result obtained by the Double Snakes algorithm.



(a) $\mathbf{y}_2^T E \mathbf{y}_1$ for $i = 1, \ldots, n$

(b) $\left\|\mathbf{y}_2^T E \mathbf{y}_1\right\|_2$

Figure 4.4: Local values of $|\mathbf{y}_2^T E \mathbf{y}_1|$ at every point on the contour as a function of time (left), and the evolution of the global norm $\|\mathbf{y}_2^T E \mathbf{y}_1\|$ during the time steps of the Double Snake algorithm (right).

4.3. Possible extensions

For the points to align more easily, we could let the first contour move as well. This leads to the coupling (via the co-planarity term) between the equations for $\mathbf{y}_1$ and $\mathbf{y}_2$, which, therefore, have to be updated simultaneously in each Forward-Euler step:

$$\frac{\partial \mathbf{y}_1}{\partial t} = \alpha \frac{\partial^2 \mathbf{y}_1}{\partial s^2} - \beta \frac{\partial^4 \mathbf{y}_1}{\partial s^4} - E^T \mathbf{y}_2 \mathbf{y}_2^T E \mathbf{y}_1 + \mathbf{F}(\mathbf{y}_1), \tag{4.9}$$

$$\frac{\partial \mathbf{y}_2}{\partial t} = \alpha \frac{\partial^2 \mathbf{y}_2}{\partial s^2} - \beta \frac{\partial^4 \mathbf{y}_2}{\partial s^4} - E \mathbf{y}_1 \mathbf{y}_1^T E^T \mathbf{y}_2 + \mathbf{F}(\mathbf{y}_2). \tag{4.10}$$

We have not performed any numerical experiments with these equations yet.

We could also update the essential matrix E during the contours' evolution. To derive the corresponding Euler-Lagrange equations, we first rewrite $\mathbf{y}_2^T E \mathbf{y}_1$ as:

$$\mathbf{y}_2^T E \mathbf{y}_1 = \left(\mathbf{y}_2^T \otimes \mathbf{y}_1^T\right) \mathbf{e} = \left(\mathbf{y}_2 \otimes \mathbf{y}_1\right)^T \mathbf{e}, \tag{4.11}$$

where

$$\mathbf{e} = \mathrm{vec}(E^T) = \mathrm{vec}\left(\begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix}^T\right) = \begin{bmatrix} e_{11} \\ e_{12} \\ e_{13} \\ e_{21} \\ e_{22} \\ e_{23} \\ e_{31} \\ e_{32} \\ e_{33} \end{bmatrix}. \tag{4.12}$$

Then, the part of the functional (4.1) involving E can be written as:

$$\begin{aligned} \Phi[\mathbf{e}] &= \frac{1}{2}\int_0^1 \left(\left(\mathbf{y}_2 \otimes \mathbf{y}_1\right)^T \mathbf{e}\right)^2 ds + \frac{1}{2}\left(\|\mathbf{e}\|_2^2 - 2\right)^2 \\ &= \frac{1}{2}\int_0^1 \mathbf{e}^T \left(\mathbf{y}_2 \otimes \mathbf{y}_1\right)\left(\mathbf{y}_2 \otimes \mathbf{y}_1\right)^T \mathbf{e}\, ds + \frac{1}{2}\left(\|\mathbf{e}\|_2^2 - 2\right)^2 \\ &= \frac{1}{2}\mathbf{e}^T \left[\int_0^1 \left(\mathbf{y}_2 \otimes \mathbf{y}_1\right)\left(\mathbf{y}_2 \otimes \mathbf{y}_1\right)^T ds\right] \mathbf{e} + \frac{1}{2}\left(\|\mathbf{e}\|_2^2 - 2\right)^2 \\ &= \frac{1}{2}\mathbf{e}^T Y \mathbf{e} + \frac{1}{2}\left(\|\mathbf{e}\|_2^2 - 2\right)^2, \end{aligned} \tag{4.13}$$

where

$$Y = \int_0^1 \left(\mathbf{y}_2 \otimes \mathbf{y}_1\right)\left(\mathbf{y}_2 \otimes \mathbf{y}_1\right)^T ds \quad \text{and} \quad Y^T = Y. \tag{4.14}$$

Computing the variational derivative:

$$\begin{aligned} \frac{d}{d\epsilon}\Phi[\mathbf{e} + \epsilon\mathbf{v}]\Big|_{\epsilon=0} &= \frac{d}{d\epsilon}\left[\frac{1}{2}(\mathbf{e} + \epsilon\mathbf{v})^T Y (\mathbf{e} + \epsilon\mathbf{v}) + \frac{1}{2}\left(\|\mathbf{e} + \epsilon\mathbf{v}\|_2^2 - 2\right)^2\right]_{\epsilon=0} \\ &= \mathbf{e}^T Y \mathbf{v} + 2\left(\|\mathbf{e}\|_2^2 - 2\right)\mathbf{e}^T \mathbf{v}, \end{aligned} \tag{4.15}$$

we obtain the following Euler-Lagrange equation for $\mathbf{e}$:

$$Y\mathbf{e} + 2\left(\|\mathbf{e}\|_2^2 - 2\right)\mathbf{e} = \mathbf{0}. \tag{4.16}$$

Unfortunately, this equation also has the trivial solution $\mathbf{e} = \mathbf{0}$, while this is clearly not the minimum of the functional $\Phi[\mathbf{e}]$. Therefore, other solutions have to be thought of to update the essential matrix E in every time step.

Numerical experiments presented in this Chapter demonstrate that the Double Snakes algorithm in its present form does not lead to the expected proper alignment of points in the second contour. We suspect that the main reason is the lack of strict convexity of the coplanarity term in the functional (4.1). This term should be studied in more detail both theoretically and numerically and a more appropriate, preferably, strictly-convex, constraint should be imposed.

5

Conclusion

The three-dimensional structure of a plant gives valuable insight into the growth process. Therefore, multi-angle time-lapse images are collected and combined in such a way that three-dimensional information becomes indirectly available to the observer.
To recover this information, in the first Chapter the GVF-Snakes algorithm was applied to detect the outer edge (outline) of a plant in several multi-angle images. A series of pre-processing steps based on color specifics of the image has been developed, including the placing of an initial contour, that ensures the success of the GVF-Snakes algorithm in detecting the plant's outline. This approach was applied to multiple images taken from different angles and at different times during the plant growth.

In the second Chapter the classical subjects of photogrammetry are considered such as the pinhole camera model and the essential matrix. After a RANSAC-based approach, point correspondences can be found in the reconstructed two-dimensional contours from which the essential matrix is estimated. The essential matrix plays a fundamental role in the relation between the camera reference frames, as it can be decomposed as the rotation and translation between the two cameras. This decomposition can be found with the help of the Singular Value Decomposition. With translation and rotation at hand and employing the projective relation between the object point and the image point, the depth information can be obtained, optimal in the least-squares sense. Three-dimensional contour reconstructions have been produced that show size, height, and some details of leaf structure of the plant. Furthermore, the three-dimensional reconstructions based on subsequent time-lapse images, give not only the three-dimensional information about the plant itself, but also information about its growth in time. To enable such dynamic observation a separate procedure for the alignment and scaling of time-lapse three-dimensional images was developed.

The estimation and the decomposition of the essential matrix is the most sensitive part of the contour reconstruction procedure, as errors in point correspondences can lead to a totally different, sometimes a severely wrong, result. While the Singular Value Decomposition, which is at the heart of the procedure, usually performs satisfactory, the result is still very sensitive to the small perturbations in the matched point coordinates. Therefore, stabilizing methods or other techniques that avoid SVD could be explored in the following research projects.

In the final Chapter we outline the first steps for a Double Snakes algorithm, where ideally we want to evolve two contours in two images simultaneously. The Euler-Lagrange equation for the essential matrix is quite straightforward, but a fully consistent updating scheme is not yet forthcoming. However, considering the relatively short time spent on the Double Snakes method and its perceived potential, this subject remains a research of interest.

Bibliography

[1] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331, 1988.

[2] Chenyang Xu and Jerry L Prince. Gradient vector flow: A new external force for snakes. In *Proceedings of IEEE computer society conference on computer vision and pattern recognition*, pages 66–71. IEEE, 1997.

[3] Duncan den Bakker. Analysis of microscopic images: A gradient vector flow based approach. *Repository TU Delft*, 2018.

[4] Richard I Hartley. In defense of the eight-point algorithm. *IEEE Transactions on pattern analysis and machine intelligence*, 19(6):580–593, 1997.

[5] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[6] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, second edition, 1998.

[7] Roger Y Tsai and Thomas S Huang. Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces. *IEEE Transactions on pattern analysis and machine intelligence*, (1):13–27, 1984.

[8] George Terzakis. Relative camera pose recovery and scene reconstruction with the essential matrix in a nutshell. Technical report, Technical report MIDAS. SNMSE. 2013. TR. 007, Marine and Industrial Dynamic, 2013.

[9] H Christopher Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133–135, 1981.

[10] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[11] Berthold KP Horn. Recovering baseline and orientation from essential matrix. *J. Opt. Soc. Am*, 110, 1990.