Universität Münster

Fachbereich 10
Mathematik und
Informatik

Master's Thesis

Cell Localization in PET with Optimal Transport

Maibrit Becker
Matr. 451523

Supervisor: Prof. Dr. Benedikt Wirth
Assisting Supervisor: Dr. Frank Wübbeling

Faculty of Mathematics and Computer Science, University of Münster, Germany
March 27, 2025

Contents

1 Introduction

In recent years, Positron Emission Tomography (PET) has become an important tool in medical imaging because it allows the detection and localization of cellular activity within the human body. PET generates one-dimensional projection data (also referred to as sinograms) by measuring emission lines along different paths through the body. Reconstructing a 3D image from these projections is a complex inverse problem. The core challenge is to compute the distribution of active cells or structures from these noisy and incomplete measurements.
In this work, I propose a method for approximating the positions of individual cells using optimal transport theory and slicing methods. The essential idea is to model the PET data as a set of randomly placed measurement points X along the measurement lines, and to represent the candidate cell positions by another set of points Y, distributed randomly in the 3D space with an associated mass $\mathbf{c}$. The goal is to iteratively improve the positions of the points in X and Y, so that they represent the ground truth of the cells, which generated the PET measurements.

The foundation of my method is inspired by the "Sliced Optimal Partial Transport" (SOPT) algorithm [1], where the authors present an approach to aligning point clouds using optimal transport theory. Specifically, in their "Algorithm 3", the authors apply their one-dimensional SOPT-algorithm to point cloud registration with slicing, a concept that I adapt to the PET reconstruction problem. Initially, the three-dimensional projections are sliced to simplify the problem into one dimension, and a correspondence between the X and Y points is computed based on the optimal transport cost. Following this, the points in Y are shifted towards the average positions of their corresponding points in X, improving the approximation. Once this alignment in 1D space is completed, a backprojection is performed to map the points back into the 3D space.

As an initialization step, filtered backprojection is used. This initial guess allows for a more accurate starting position of the candidate cells in the 3D space, which can then be improved iteratively using the SOPT algorithm and the idea presented above. The

process repeats iteratively, where each step consists of moving the points X toward the reconstructed points Y along the measurement lines, gradually converging to an accurate 3D representation of the cell distribution.

I also give a theoretical proof of local convergence for the proposed algorithm, when starting near the ground truth. By proving that my algorithm operates similarly to the minimization of a sliced Wasserstein distance between the sets of points X and Y using the stochastic gradient descent algorithm, the convergence can be shown.
I implemented my algorithm in Python through numerical experiments with practical examples and investigated the algorithms' results while testing different parameters.

In the following sections, a detailed mathematical formulation of the problem will be provided, the derivation of the algorithm will be explained and the numerical results of the algorithm will be presented.

2 Optimal Transport: Mathematical Basics

This chapter provides the necessary mathematical foundations to understand the algorithms and techniques introduced in this work. The main idea is based in the theory of Optimal Transport, especially with a focus on the ideas introduced in the Sliced Optimal Partial Transport article [1]. Optimal Transport is a mathematical theory that deals with finding the most efficient way to move mass from one distribution to another, minimizing a given cost function.
We will begin by introducing the classical Kantorovich formulation of the optimal transport problem, followed by its dual formulation, and derived from that its optimality conditions. Special attention will be given to transport scenarios where the source and target measures have different total masses.
Finally, we will introduce the concept of slicing, a technique that allows high-dimensional transport problems to be reduced to simpler, lower-dimensional cases.

2.1 Optimal Transport

To stay consistent throughout the thesis with the formulations in the SOPT-articel [1], the definitions and statements in this chapter are formulated in line with those in the article.

Definition 2.1 (Measure)**.** Let X be a set and $\mathcal{A}$ a σ-algebra over X. A function $\mu : \mathcal{A} \to [0, \infty]$ is called a measure if the following conditions hold:

- $\mu(\emptyset) = 0$ and
- for all countable collections $\{A_k\}_{k=1}^{\infty}$ of pairwise disjoint sets in $\mathcal{A}$ holds: $\mu(\cup_{k=1}^{\infty} A_k) = \sum_{k=1}^{\infty} \mu(A_k)$ (σ-additivity).

A probability measure is a measure, where $\mu(X) = 1$ holds.

Example 2.2 (Discrete Measure)**.** A discrete measure μ on $(X, \mathcal{A})$ with locations $x_1, ..., x_n \in X$ reads

$$\mu = \sum_{i=1}^{\infty} a_i \delta_{x_i},$$

where the weights $a_i \in \mathbb{R}_{>0}$ and Dirac measures $\delta_{x_i}(X)$ defined as

$$\delta_{x_i}(X) = \begin{cases} 1 \text{ if } x_i \in X \\ 0 \text{ if } x_i \notin X. \end{cases}$$

Definition 2.3 (Push Forward measure)**.** Given a measurable function $f : X \to Y$ between two measurable spaces and a measure μ on X, the pushforward measure $f_{\#}\mu$ on Y is defined by

$$(f_{\#}\mu)(B) = \mu(f^{-1}(B)),$$

for any measurable set $B \subseteq Y$.

Definition 2.4 (Wasserstein-distance)**.** Let (Ω, d) be a metric space, where every probability measure is a radon measure in Ω.
For $p \geq 1$ let $\mathcal{P}_p(\Omega)$ the set of all probability measures μ on Ω with finite p'th Moment, meaning for a x_0 from Ω holds

$$\int_\Omega d(x, x_0)^p d\mu(x) < \infty.$$

Then the Wasserstein p-distance between two probability measures μ and ν from $\mathcal{P}_p(\Omega)$ for $p < \infty$ is defined as

$$W_p(\mu, \nu) := \left(\inf_{\gamma \in \Gamma(\mu,\nu)} \int_{\Omega \times \Omega} d(x, y)^p d\gamma(x, y) \right)^{\frac{1}{p}},$$

where $\Gamma(\mu, \nu)$ is the set of all joint probability measures whose marginal are μ and ν, meaning

$$\Gamma(\mu, \nu) = \{\gamma : \pi_{1\#}\gamma = \mu, \pi_{2\#}\gamma = \nu\},$$

where $\pi_1(x, y) = x$ and $\pi_2(x, y) = y$. This is also called "mass conservation constraint".
For $p = \infty$ the Wassertein distance is defined as

$$W_\infty(\mu, \nu) := \inf_{\gamma \in \Gamma(\mu,\nu)} \sup_{(x,y) \in \text{supp}(\gamma)} d(x, y),$$

where $\text{supp}(\gamma)$ is the support of the measure.

Remark 2.5. The Wasserstein-distance is a metric on $\mathcal{P}_p(\Omega)$.

Proof. See for example [7], Chapter 6, Example 6.3.

□

To develop their algorithm the authors from [1] used the Kantorovich definition of the Optimal (partial) transport problem, which can be derived from the Wasserstein-metric. In the following, we will use their definition of the Optimal Transport problem:

Definition 2.6 (Kantorovich Problem)**.** Given $\mu, \nu \in \mathcal{P}(\Omega)$, and a lower semi-continuous function $c : \Omega^2 \to \mathbb{R}_+$, the OT problem between μ and ν in the Kantorovich formulation is defined as

$$\mathrm{OT}(\mu, \nu) := \inf_{\gamma \in \Gamma(\mu,\nu)} \int_{\Omega^2} c(x, y)\, d\gamma(x, y). \tag{2.1}$$

When $c(x, y)$ is the p-th power of a metric, the p-th root of the induced optimal value is the Wasserstein distance.

Remark 2.7. Suppose we have empirical distributions μ and ν with the same sizes and weights = 1:
$\mu = \sum_{i=1}^{n} \delta_{x_i}$ and $\nu = \sum_{j=1}^{n} \delta_{y_j}$ with $n = m$. Then the OT problem 2.1 for those measures can be written as

$$\mathrm{OT}(\{x_i\}_{i=1}^{n}, \{y_j\}_{j=1}^{m}) := \min_{\gamma \in \Gamma(\mu,\nu)} \sum_{i,j} c(x_i, y_j)\gamma_{ij}, \tag{2.2}$$

where

$$\Gamma(\mu, \nu) := \left\{\gamma \in \mathbb{R}_+^{n\times m} : \gamma \mathbf{1}_m = \mathbf{1}_n,\ \gamma^T \mathbf{1}_n = \mathbf{1}_m\right\}$$

and $\mathbf{1}_n$ denotes the $n \times 1$ vector whose entries are 1 and analogously for $\mathbf{1}_m$.

Remark 2.8 (Dual of the empirical OT-problem)**.** The primal problem 2.2 admits the dual form

$$\sup_{\substack{\Phi\in\mathbb{R}^n, \Psi\in\mathbb{R}^m \\ \Phi_i+\Psi_j \le c(x_i,y_j)\,\forall i,j}} \sum_{i=1}^{n} \Phi_i + \sum_{j=1}^{m} \Psi_j.$$

Proof.
Let's denote $C_{ij} := c(x_i, y_j)$. Then we can write $\langle C, \gamma\rangle = \sum_{i,j} c(x_i, y_j)\gamma_{ij}$.
Next, we write down the Lagrangian for the OT problem (while keeping in mind the definition of $\Gamma(\mu, \nu)$:

$$\min_{\gamma} \max_{(\Phi,\Psi)\in(\mathbb{R}^n\times\mathbb{R}^m)} \langle C, \gamma\rangle + \langle \mathbf{1}_n - \gamma\mathbf{1}_n, \Phi\rangle + \langle \mathbf{1}_m - \gamma^T\mathbf{1}_m, \Psi\rangle.$$

Next, we exchange min and max. This is always possible when considering linear programs in finite dimensions. Then we get

$$\max_{(\Phi,\Psi)\in(\mathbb{R}^n\times\mathbb{R}^m)} \langle \mathbf{1}_n, \Phi\rangle + \langle \mathbf{1}_m, \Psi\rangle + \min_{\gamma} \langle C, \gamma\rangle - \langle \gamma \mathbf{1}_n, \Phi\rangle - \langle \gamma^T \mathbf{1}_m, \Psi\rangle$$

$$= \max_{(\Phi,\Psi)\in(\mathbb{R}^n\times\mathbb{R}^m)} \langle \mathbf{1}_n, \Phi\rangle + \langle \mathbf{1}_m, \Psi\rangle + \min_{\gamma} \langle \gamma, \underbrace{C - \Phi \mathbf{1}_n^T - \Psi^T \mathbf{1}_m}_{:=Q}\rangle$$

$$\min_{\gamma\geq 0} \langle \gamma, Q\rangle = \begin{cases} 0 & \text{if } Q \geq 0, \\ -\infty & \text{otherwise.} \end{cases}$$

Because we want to calculate the maximum of $\min\langle \gamma, Q\rangle$, and this is 0, we get the constraint:

$$C - \Phi \mathbf{1}_n^T - \Psi^T \mathbf{1}_m = C - \Phi \oplus \Psi > 0.$$

□

Example 2.9. An interesting example to understand the primal/dual formulation better is provided in [5], Remark 2.10 and Remark 2.21. In the following, this idea is presented:
The primal Kantorovich optimal transport problem can be practically interpreted through the example of resource allocation between warehouses and factories. In this scenario, an operator wants to transport raw materials from n warehouses to m factories at minimal cost. Each warehouse x_i holds units of resources, while each factory y_j requires units to operate. The cost of transporting one unit from warehouse x_i to factory y_j is denoted by $c(x_i, y_j)$.
The objective is to find an optimal transport plan γ_{ij} that minimizes the total cost:

$$\min_{\gamma\in\Gamma(\mu,\nu)} \sum_{i,j} c(x_i, y_j)\gamma_{ij},$$

which represents the primal problem.
$\Gamma(\mu, \nu)$ represents the set of feasible transport plans that satisfy the supply and demand constraints, so that all units from all factories get transported and that each factory gets all resources it requires.

The dual problem can be described as follows:

The operator gets the opportunity to outsource the transport task to a vendor. The vendor employs a pricing scheme, where they charge a collection fee Φ_i for picking up a unit from warehouse x_i and a delivery fee Ψ_j for delivering a unit to factory y_j. Thus, the total price charged by the vendor is:

$$\sum_{i=1}^{n} \Phi_i + \sum_{j=1}^{m} \Psi_j. \tag{2.3}$$

To ensure competitive pricing, the vendor must set prices such that:

$$\Phi_i + \Psi_j \leq c(x_i, y_j) \quad \text{for all } i, j.$$

This means the vendor has to make sure, that their prices do not exceed the operators transporting cost, otherwise their offer would not be accepted.

The vendor's objective is to maximize their profit in 2.3 while sticking to these cost constraints. Therefore, the dual problem is then defined as the maximum over their profit 2.3, while keeping in mind to keep their prices lower than the operators transporting costs:

$$\sup_{\Phi,\Psi} \left(\sum_{i=1}^{n} \Phi_i + \sum_{j=1}^{m} \Psi_j \right),$$

subject to the condition $\Phi_i + \Psi_j \leq c(x_i, y_j)$.
The optimal dual solution corresponds to the maximum price the vendor can charge while still being competitive with the direct transportation cost.
Note that the optimal value of the dual problem is equal to that of the primal problem, indicating that the price charged by the vendor matches the minimal transport cost the operator would have if they solved the problem themselves. This duality provides a quick way for the operator to check the fairness of the vendor's prices, ensuring that they do not overpay for the service.

Theorem 2.10. Let K, T be convex subsets of vector spaces V_1, respectively V_2, where V_1 is locally convex and let $f : K \times T \to \mathbb{R}$. If

1. $f(x, \cdot)$ is concave for every $x \in K$
2. $f(\cdot, y)$ is continuous and convex on K for every $y \in T$
3. $f(x, \cdot)$ is concave on T for every $x \in K$

then

$$\sup_{y\in T}\inf_{x\in K} f(x,y) = \inf_{x\in K}\sup_{y\in T} f(x,y).$$

PROOF. See [2], Theorem 2.

□

2.2 Optimal Partial Transport

In many use cases, it is relevant to look at measures with different amounts of masses. Because of the mass conservation constraint, we can not use the OT problem. Instead, in [1] the authors modified the OT problem to allow mass destruction on the source and mass creation on the target by introducing a linear mass destruction and creation penalty.

Definition 2.11 (Kantorovich Problem for Optimal Partial Transport)**.** Let $\mu, \nu \in \mathcal{M}_+(\Omega)$, the set of all positive Radon measures on Ω and $\lambda_1, \lambda_2 \geq 0$. The definition of the OPT Problem from [1] is:

$$\mathrm{OPT}_{\lambda_1,\lambda_2}(\mu,\nu) := \inf_{\gamma\in\Gamma_{\leq}(\mu,\nu)} \int_{\Omega^2} c(x,y)\, d\gamma + \lambda_1 \left(\mu(\Omega) - \pi_{1\#}\gamma(\Omega)\right) + \lambda_2 \left(\nu(\Omega) - \pi_{2\#}\gamma(\Omega)\right), \tag{2.4}$$

where

$$\Gamma_{\leq}(\mu,\nu) = \{\gamma : \pi_{1\#}\gamma \leq \mu, \pi_{2\#}\gamma \leq \nu\}.$$

Partial transport means that we transport a fixed amount of mass $m = \gamma(\Omega)^2$ between the two measures μ and ν with different masses as cheap as possible.
At the right hand side of equation 2.4 the penalty term was added which gets larger, the less we transport. Without the penalty term, due to the definition of $\Gamma_{\leq}(\mu,\nu)$, no transport at all would be the optimal value of our OPT-problem.

Remark 2.12. The authors of [1] also stated that, without loss of generality, it is possible to consider for $\lambda = \frac{\lambda_1+\lambda_2}{2}$ only

$$\mathrm{OPT}_{\lambda}(\mu,\nu) := \mathrm{OPT}_{\lambda,\lambda}(\mu,\nu),$$

because

$$\mathrm{OPT}_{\lambda,\lambda}(\mu,\nu) = \mathrm{OPT}_{\lambda_1,\lambda_2}(\mu,\nu) \underbrace{-\frac{\lambda_1-\lambda_2}{2}\mu(\Omega) - \frac{\lambda_2-\lambda_1}{2}\nu(\Omega)}_{=:K_{\lambda_1,\lambda_2}(\mu,\nu)}$$

and $K_{\lambda_1,\lambda_2}(\mu,\nu)$ is a constant for fixed μ and ν.

Remark 2.13. When $c(x,y)$ is a metric and $\lambda_1 = \lambda_2$, then $\mathrm{OPT}_\lambda(.,.)$ defines a metric on $\mathcal{M}_+(\Omega)$

Proof. For the details, again, look at [1], Page 3.

□

One question left is the following: How does the choice of λ affect the result of our OPT-Problem?
Shortly said, λ corresponds to the maximum distance mass can be transported. Each amount of mass m can be achieved by choosing λ: The bigger we choose it, the bigger the transported mass. It is also possible to see that in equation 2.4: The variable λ influences the impact of the penalty term.
Next, the authors of [1] formalized this result by showing that the support of the optimal transport plan γ does not contain pairs of points (x,y) where the cost to transport x to y exceeds 2λ:

Lemma 2.14. There exists an optimal γ^* for (3) such that $\gamma^*(S) = 0$, where $S = \{(x,y) \in \Omega^2 : c(x,y) \geq 2\lambda\}$.

Proof. The proof follows along the proof in [1].
Let γ be any transport plan, and define a new transport plan γ' by

$$\gamma'(A) = \gamma(A \setminus S)$$

for any Borel set $A \subset \Omega^2$. The corresponding cost functional $C(\gamma)$ is given by:

$$C(\gamma) = \int_{\Omega^2} c(x,y)\, d\gamma + \lambda \left[\mu(\Omega) - \pi_{1\#}\gamma(\Omega) + \nu(\Omega) - \pi_{2\#}\gamma(\Omega)\right].$$

Since $\gamma(\Omega^2) = (\pi_{1\#}\gamma)(\Omega) = (\pi_{2\#}\gamma)(\Omega)$, we can rewrite the cost functional as:

$$C(\gamma) = \int_{\Omega^2} (c(x,y) - 2\lambda)\, d\gamma + \lambda(\mu(\Omega) + \nu(\Omega)).$$

The difference in cost between γ and γ' is then:

$$C(\gamma) - C(\gamma') = \int_S (c(x,y) - 2\lambda)\, d\gamma(x,y).$$

Since $c(x,y) \geq 2\lambda$ for all $(x,y) \in S$ by the definition of S, it follows that:

$$C(\gamma) - C(\gamma') \geq 0.$$

This implies that for any γ, we can find a better transport plan γ' such that $\gamma'(S) = 0$.
□

2.3 Dual and optimality conditions

The proofs in this section essentially follow the proofs presented in [4].

Theorem 2.15 (Dual)**.** Define the Csiszár f–divergence $\mathcal{F}(\gamma_1, \mu)$ as

$$\mathcal{F}(\gamma_1, \mu) := \int F(\sigma) d\mu + F'_\infty \gamma^\perp(\Omega) = \begin{cases} \lambda\,(\mu(\Omega) - \gamma_1(\Omega)) & \text{if } 0 \leq \gamma_1 \leq \mu, \\ +\infty & \text{otherwise,} \end{cases}$$

(same for $\mathcal{F}(\gamma_2, \nu)$)
with the integrand F

$$F(s) = \begin{cases} \lambda(1-s) & \text{if } s \in [0,1], \\ +\infty & \text{else.} \end{cases}$$

and $\sigma, \mu^\perp$ is defined by Lebesgue's decomposition theorem $\mu = \frac{d\gamma_1}{d\mu} + \mu^\perp$ and recession constant of F: $F'_\infty := \lim_{s\to\infty} \frac{F(s)}{s}$.
Then we can rewrite our OPT-problem 2.4 by plugging everything in by:

$$\mathrm{OPT}_\lambda(\mu,\nu) = \inf_{\gamma \geq 0} \int_{\Omega^2} c\, d\gamma + \mathcal{F}(\underbrace{\pi_{1\#}\gamma}_{=:\gamma_1}, \mu) + \mathcal{F}(\underbrace{\pi_{2\#}\gamma}_{=:\gamma_2}, \nu). \tag{2.5}$$

We show: The dual of our OPT-problem in 2.4 is

$$\mathcal{D}(\mu,\nu) = \sup_{\substack{\Phi \in L^1(\mu),\, \Psi \in L^1(\nu) \\ \Phi \oplus \Psi \leq c \\ \Phi, \Psi \text{ lsc and bounded}}} \int_\Omega -F^*(-\Phi)\, d\mu + \int_\Omega -F^*(-\Psi)\, d\nu,$$

where $F^* : \mathbb{R} \to (-\infty, \infty]$, called Legendre conjugate function, is defined as

$$F^*(r) = \sup_s(rs - F(s)) = \max\{-\lambda, r\}.$$

PROOF. By [4], Theorem 2.7 one can rewrite

$$\mathcal{F}(\gamma_1, \mu) = \sup\left\{\int_\Omega \Phi d\mu - \int_\Omega R^*(\Phi)\, d\gamma_1 : \Phi, R^*(\Phi) \text{ lower semicontinuous (lsc) and bounded}\right\},$$

$$\text{where } R^*(\Phi) = \sup_{s>0}(s\Phi - sF(1/s)) = \sup_{r>0}(\Phi - F(r))/r$$

and

$$\mathcal{F}(\gamma_2, \nu) = \sup\left\{\int_\Omega \Psi d\nu - \int_\Omega R^*(\Psi)\, d\gamma_2 : \Psi, R^*(\Psi) \text{ lower semicontinuous and bounded}\right\},$$

$$\text{where } R^*(\Psi) = \sup_{s>0}(s\Psi - sF(1/s)) = \sup_{r>0}(\Psi - F(r))/r.$$

Next, we introduce the saddle function

$$\mathcal{L}(\gamma, (\Phi, \Psi)) = \underbrace{\int_\Omega c - R^*(\Phi) - R^*(\Psi)\, d\gamma}_{I} + \underbrace{\int_\Omega \Phi\, d\mu + \int_\Omega \Psi\, d\nu}_{II}. \tag{2.6}$$

By rewriting our OPT-problem with our new $\mathcal{F}(\gamma_1, \mu)$, $\mathcal{F}(\gamma_2, \nu)$ we immediately see that

$$\mathrm{OPT}_\lambda(\mu, \nu) = \inf_{\gamma>0} \sup_{\substack{\Phi, R^*(\Phi), \\ \Psi, R^*(\Psi) \\ \text{lsc and bounded}}} \mathcal{L}(\gamma, (\Phi, \Psi)).$$

The idea is to get the dual by exchanging inf and sup. One can show that

$$\sup \inf_\gamma \mathcal{L}(\gamma, \varphi) = \inf_\gamma \sup \mathcal{L}(\gamma, \varphi),$$

using Theorem 2.10.
For our saddle function $\mathcal{L}(\gamma, (\Phi, \Psi))$ one can show:

1. $\gamma \mapsto \mathcal{L}(\gamma, (\Phi, \Psi))$ is convex and lsc.
2. $(\Phi, \Psi) \mapsto \mathcal{L}(\gamma, (\Phi, \Psi))$ is concave.
3. There exist (Φ_0, Ψ_0) and $c > \sup \inf \mathcal{L}(\gamma, (\Phi, \Psi))$ s.t. $\{\gamma : \mathcal{L}(\gamma, (\Phi_0, \Psi_0)) \leq c\}$ is compact.

(Again, see [4], Theorem 2.4 for details). Then we obtain the desired equality and hence define the dual of OPT as:

$$\mathcal{D}(\mu,\nu) = \sup_{\substack{\Phi,R^*(\Phi),\\ \Psi,R^*(\Psi)\\ \text{lsc and bounded}}} \inf_{\gamma>0} \mathcal{L}\left(\gamma,(\Phi,\Psi)\right).$$

Next, we calculate $\inf_{\gamma>0}\mathcal{L}\left(\gamma,(\Phi,\Psi)\right)$. For Part I from 2.6 we get

$$\inf_{\gamma>0}\int_\Omega c - R^*(\Phi) - R^*(\Psi)d\gamma = \begin{cases} 0 & \text{if } R^*(\Phi)+R^*(\Psi)\le c \\ -\infty & \text{else.}\end{cases}$$

Then, also considering Part II from 2.6, this leads to:

$$\inf_{\gamma>0}\mathcal{L}\left(\gamma,(\Phi,\Psi)\right) = \begin{cases} \int_\Omega \Phi\, d\mu + \int_\Omega \Psi\, d\nu & \text{if } R^*(\Phi)+R^*(\Psi)\le c \\ -\infty & \text{else,}\end{cases}$$

so the dual has the form:

$$\mathcal{D}(\mu,\nu) = \sup_{\substack{\Phi,R^*(\Phi),\Psi,R^*(\Psi)\\ \text{lsc and bounded}\\ R^*(\Phi)\oplus R^*(\Psi)\le c}} \int_\Omega \Phi d\mu + \int_\Omega \Psi d\nu.$$

By change of variable: $\varphi_1 := -R^*(\Phi)$, $\varphi_2 := -R^*(\Psi)$, $\Phi := -F^*(-\varphi_1)$ and $\Psi := -F^*(-\varphi_2)$ we get

$$\mathcal{D}(\mu,\nu) = \sup_{\substack{\varphi_1\in L^1(\mu),\varphi_2\in L^1(\nu)\\ \varphi_1,\varphi_2 \text{ lsc and bounded}\\ \phi_1\oplus\phi_2\le c}} \int_\Omega -F^*(-\varphi_1)d\mu + \int_\Omega -F^*(-\varphi_2)d\nu,$$

which is what we claimed with $\varphi_1=\Phi$, $\varphi_2=\Psi$.

□

Remark 2.16. Again, as in Remark 2.7, suppose we have our two discrete measures μ and ν in $\mathbb{R}^d$, but now with different sizes, so $n\neq m$. Then we define the discrete version of our OPT-Problem 2.4 by

$$\mathrm{OPT}(\{x_i\}_{i=1}^n,\{y_j\}_{j=1}^m) := \min_{\gamma\in\Gamma\le(\mu,\nu)} \sum_{i,j} c(x_i,y_j)\gamma_{ij} + \lambda(n+m-2\sum_{i,j}\gamma_{ij}), \qquad (2.7)$$

where

$$\Gamma_{\leq}(\mu,\nu) := \left\{\gamma \in \mathbb{R}_{+}^{n\times m} : \gamma \mathbf{1}_m \leq \mathbf{1}_n,\ \gamma^T \mathbf{1}_n \leq \mathbf{1}_m\right\}$$

and $\mathbf{1}_n$ denotes the $n \times 1$ vector whose entries are 1.

Remark 2.17 (Dual of the discrete OPT-problem)**.** The discrete primal problem 2.7 admits the dual form

$$\sup_{\substack{\Phi\in\mathbb{R}^n,\Psi\in\mathbb{R}^m \\ \Phi_i+\Psi_j\leq c(x_i,y_j)\,\forall i,j}} \sum_{i=1}^{n} \min\{\Phi_i,\lambda\} + \sum_{j=1}^{m} \min\{\Psi_j,\lambda\}.$$

Theorem 2.18 (Optimality Conditions)**.** We have the following necessary and sufficient conditions for $\gamma \in \Gamma_{\leq}(\mu,\nu), \Phi \in L^1(\mu),\ \Psi \in L^1(\nu)$ to be optimal for the primal and dual problems:

$$\Phi \oplus \Psi = c \quad \gamma\text{-a.e.}$$

$$-\Phi \in \partial F\left(\frac{d\gamma_1}{d\mu}\right), \quad \gamma_1 = \pi_1 \# \gamma \quad \mu\text{-a.e.}$$

$$-\Psi \in \partial F\left(\frac{d\gamma_2}{d\nu}\right), \quad \gamma_2 = \pi_2 \# \gamma \quad \nu\text{-a.e.}$$

Proof. It is possible to show that:

$$\mathcal{F}(\gamma_1,\mu) - \int_\Omega \Phi\, d\mu \geq \int_\Omega \phi_1\, d\gamma_1$$

and that equality holds if and only if for the Lebesgue decomposition, one has:

$$\begin{cases} \phi_1 \in \partial F(\frac{\partial\gamma_1}{\partial\mu}), \quad \Phi = -F^*(\phi_1) & \mu+\gamma_1\text{-a.e.} \\ \Phi = F(0) < \infty & \mu^{\perp}\text{-a.e.} \\ \phi_1 = F'_\infty < \infty & \gamma_1^{\perp}\text{-a.e.} \end{cases}$$

Analogue for

$$\mathcal{F}(\gamma_2,\nu) - \int_\Omega \Psi\, d\nu \geq \int_\Omega \phi_2\, d\gamma_2,$$

with ∂F the subdifferential of F (See again [4], Lemma 2.6 for the proof).
We again substitute $\Phi := -F^*(-\varphi_1)$, $\phi_1 = -\varphi_1$ (and analogue $\Psi := -F^*(-\varphi_2)$, $\phi_2 = -\varphi_2$) to obtain

$$\mathcal{F}(\gamma_1,\mu) - \int_\Omega -F^*(-\varphi_1) d\mu \geq \int_\Omega -\varphi_1\, d\gamma_1, \tag{2.8}$$

with equality if and only if:

$$\begin{cases} -\varphi_1 \in \partial\mathcal{F}(\frac{\partial\gamma_1}{\partial\mu}), \quad -F^*(-\varphi_1) = -F^*(\varphi) & (\mu+\gamma_1)\text{-a.e.} \\ -F^*(-\varphi_1) = F(0) < \infty & \mu^\perp\text{-a.e.} \\ -\varphi_1 = F'_\infty < \infty & \gamma_1^\perp\text{-a.e.} \end{cases} \tag{2.9}$$

and analogous

$$\mathcal{F}(\gamma_2, \nu) - \int_\Omega -F^*(-\varphi_2) d\nu \geq \int_\Omega -\varphi_2 \, d\gamma_2. \tag{2.10}$$

By recalling the definition of OPT (2.5) and using 2.8, 2.10 and 2.9, we get

$$\underbrace{\underbrace{\int_{\Omega^2} c \, d\gamma}_{\geq \int_\Omega \varphi_1 d\gamma + \int_\Omega \varphi_2 d\gamma} + \mathcal{F}(\gamma_1, \mu) + \mathcal{F}(\gamma_2, \nu)}_{\textit{objective of ET}} \overset{(2.8)}{\geq} \underbrace{\int_\Omega -F^*(-\varphi_1) d\mu + \int_\Omega -F^*(-\varphi_2) d\nu}_{\text{objective of the dual}},$$

where equality holds if and only if

$$\begin{cases} c = \varphi_1 \oplus \varphi_2 & \gamma\text{-a.e.} \\ -\varphi_1 \in \partial F(\frac{\partial\gamma_1}{\partial\mu}) & \mu\text{-a.e.} \quad (*) \\ -\varphi_2 \in \partial F(\frac{\partial\gamma_2}{\partial\nu}) & \nu\text{-a.e. ,} \end{cases}$$

with

$$\partial F(s) = \begin{cases} \{-\lambda\} & \text{if } s \in (0,1) \\ (-\infty, -\lambda] & \text{if } s = 0 \\ [-\lambda, \infty) & \text{if } s = 1. \end{cases}$$

Then for example (*) is equivalent to:

$$\varphi_1(x) = \lambda \quad \text{if } \frac{\partial\gamma_1(x)}{\partial\mu(x)} \in (0,1)$$

$$\varphi_1(x) \in [\lambda, \infty) \quad \text{if } \frac{\partial\gamma_1(x)}{\partial\mu(x)} = 0$$

$$\varphi_1(x) \in (-\infty, \lambda] \quad \text{if } \frac{\partial\gamma_1(x)}{\partial\mu(x)} = 1,$$

which is again the statement we claimed with $\varphi_1 = \Psi$ and $\varphi_2 = \Phi$. □

Remark 2.19 (Optimality Conditions for the discrete case)**.** We get the following optimality conditions for the discrete case $\gamma \in \Gamma_{\leq}(\mu, \nu), \Phi \in \mathbb{R}^n, \Psi \in \mathbb{R}^m$:

$$\begin{aligned}
&\Phi_i + \Psi_j = c(x_i, y_j), \forall (x_i, y_j) \in \operatorname{supp}(\gamma) \\
&\Phi_i < \lambda \Rightarrow [\pi_{1\#}\gamma]_i = 1 \qquad \Psi_j < \lambda \Rightarrow [\pi_{2\#}\gamma]_j = 1 \\
&\Phi_i = \lambda \Rightarrow [\pi_{1\#}\gamma]_i \in [0,1] \qquad \Psi_j = \lambda \Rightarrow [\pi_{2\#}\gamma]_j \in [0,1] \\
&\Phi_i > \lambda \Rightarrow [\pi_{1\#}\gamma]_i = 0 \qquad \Psi_j > \lambda \Rightarrow [\pi_{2\#}\gamma]_j = 0
\end{aligned}$$

2.4 Slicing

In order to extend the concept of optimal transport to higher dimensions, we introduce a technique known as slicing. Slicing is a method that allows us to simplify high-dimensional problems by projecting them onto lower-dimensional subspaces, typically one-dimensional lines. By reducing the dimensionality of the problem, we can handle complex measures and distributions more easily.

To formalize this, consider a set of probability measures in a high-dimensional space, specifically in $\mathbb{R}^d$. Let $\mu, \nu \in \mathcal{M}_+(\Omega)$, where $\Omega \in \mathbb{R}^d$ be two measures and let $\lambda : S^{d-1} \to \mathbb{R}_+ \setminus \{0\}$. The core idea is to define the Sliced Optimal Partial Transport problem (like in [1]) in terms of integrals over all directions on the sphere S^{d-1}:

$$\mathrm{SOPT}_\lambda(\mu, \nu) = \int_{S^{d-1}} \mathrm{OPT}_{\lambda(\theta)} \left(\langle \theta, \cdot \rangle_{\#}\mu,\, \langle \theta, \cdot \rangle_{\#}\nu \right) d\sigma(\theta),$$

where $\sigma \in \mathcal{P}(S^{d-1})$ be a set of probability measures supported on the unit sphere S^{d-1} and $\langle \theta, \cdot \rangle_{\#}$ denotes the projection of measures onto the direction θ. For general measures, this integral is often computationally intensive, so it is replaced with an empirical average:

$$\mathrm{SOPT}_\lambda(\mu, \nu) \approx \frac{1}{N} \sum_{i=1}^{N} \mathrm{OPT}_{\lambda_i} \left(\langle \theta_i, \cdot \rangle_{\#}\mu,\, \langle \theta_i, \cdot \rangle_{\#}\nu \right),$$

where $\{\theta_i\}_{i=1}^N$ now are sampled from a uniform distribution over the unit sphere S^{d-1} and λ is an L^1-function $\lambda \in L^1(S^{d-1}; \mathbb{R}_+ \setminus \{0\})$.

In practical terms, this means that with the help of slicing, we project the high-dimensional data points X and Y onto lines defined by directions θ_i. These projections transform the points into one-dimensional distributions (essentially a sum of delta functions along the projection line). We then compute the optimal transport in this reduced one-dimensional setting for each θ_i, and average the results over multiple directions to

approximate the high-dimensional transport cost.

Theorem 2.20. Let $\lambda : S^{d-1} \to \mathbb{R}_+ \setminus \{0\}$. When the cost function $c : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ is the p-th power of a metric on $\mathbb{R}$, where $p \in [1, \infty)$, then $(\mathrm{SOPT}_\lambda(\mu, \nu))^{\frac{1}{p}}$ is a metric in $\mathcal{M}_+(\Omega)$.

Proof. The proof can be found in [1], Theorem 5.2.

□

3 Sliced Optimal Partial Transport: The Algorithm

3.1 The Sliced Optimal Partial Transport Algorithm

Algorithm 1: opt-1d

Input: $\{x_i\}_{i=1}^n, \{y_j\}_{j=1}^m, \lambda$

Output: L, Ψ, Φ

Initialize $\Phi_i \leftarrow -\infty$ for $i \in [1:n]$, $\Psi_j \leftarrow \lambda$ for $j \in [1:m]$ and $L_i \leftarrow -1$ for $i \in [1:n]$

for $k = 1, 2, \dots n$ **do**

  $j^* \leftarrow \text{argmin}_{j \in [1:m]}\, c(x_k, y_j) - \Psi_j$

  $\Phi_k \leftarrow \min\{c(x_k, y_{j^*}) - \Psi_{j^*}, \lambda\}$

  if $\Phi_k = \lambda$ **then**

    [Case 1] No update on L

  else if j^* *unassigned* **then**

    [Case 2] $L_k \leftarrow j^*$

  else

    [Case 3] Run Algorithm 2.

Figure 3.1: OPT-algorithm pseudocode from [1]

The Sliced Optimal Partial Transport (SOPT) from [1] algorithm is an iterative method designed to solve the partial optimal transport problem by projecting the problem into one-dimensional slices. Below, we describe the algorithm in detail, including the steps taken in each iteration.

Throughout the algorithm, not only the primal variable γ, but also the dual variables Φ and Ψ are iteratively adjusted. The dual objective function is designed to maximize the transport potential while respecting the penalty factor λ, the primal objective function wants to minimize the total transport cost.

Initialization and Setup

We assume that the source and target points $\{x_i\}_{i=1}^n$ and $\{y_j\}_{j=1}^m$ are sorted and that $L[i]$ is the index determining the transport of mass from x_i, which means if $L[i] \neq -1$ then x_i is assigned to $y_{L[i]}$.

The SOPT algorithm begins with an initialization step, after that we assume that the optimal transport solution has been solved up to the current point, and the goal is to extend it to the next point in the sequence. For this purpose, we have our discrete measures defined by source points x_i and target points y_j, which we aim to align optimally.
Assume we already solved

$$\mathrm{OPT}\left(\{x_i\}_{i=1}^{k-1}, \{y_j\}_{j=1}^{m}\right).$$

In the next Iteration we want to solve

$$\mathrm{OPT}\left(\{x_i\}_{i=1}^{k}, \{y_j\}_{j=1}^{m}\right).$$

In each iteration k, a new point x_k is introduced, and the algorithm determines its optimal match among the target points y_j by minimizing the cost function subject to the dual constraints.

Selection of the Most Attractive Target Point

To determine the optimal target for the new point x_k, the algorithm computes the most attractive target point y_{j^*}, which minimizes the cost function $c(x_k, y_j)$ adjusted by the dual variables Ψ and Φ. Then we calculate the index j^* of the most attractive point y_j by calculating

$$j^* = \arg\min_{j=1,\ldots,m} \left(c(x_k, y_j) - \Psi_j\right)$$

and

$$\Phi_k = \min\{c(x_k, y_{j^*}) - \Psi_{j^*}, \lambda\}.$$

The parameter λ acts as a penalty factor, controlling the maximum allowed transport cost.

Handling Different Cases

The algorithm proceeds by evaluating the value of Φ_k and then distinguishing three main cases:

- **Case 1:** If $\Phi_k = \lambda$, then x_k is considered "destroyed," meaning it does not get assigned to any y_j.
- **Case 2:** If $\Phi_k < \lambda$ and y_{j^*} is *unassigned*, then we assign x_k to y_{j^*}.

- **Case 3:** If $\Phi_k < \lambda$ and y_{j^*} is already *assigned*, we usually would also assign x_k to y_{j^*}, but because this is not possible, we start a sub-algorithm to resolve the conflict.

Conflict Resolution and Sub-Algorithm

When a conflict arises (Case 3), a sub-algorithm is started to adjust the assignments and to solve this conflict. This involves raising Φ_{k-1} and Φ_k while lowering Ψ_{j^*} at the same rate, until one of the following sub-cases occurs:

- **Case 3.1:** Φ_{k-1} (or Φ_k) reaches λ. In this situation, we assign x_k to y_{j^*}, and destroy x_{k-1} (or assign x_{k-1} to y_{j^*}, and destroy x_k).
- **Case 3.2:** $\Phi_k + \Psi_{j^*+1} = c(x_k, y_{j^*+1})$ is satisfied (which would be optimal for the primal **and** the dual variables), allowing x_k to be assigned to the new target y_{j^*+1}.
- **Case 3.3:** If $\Phi_{k-1} + \Psi_{j^*-1} = c(x_{k-1}, y_{j^*-1})$ and y_{j^*-1} is unassigned, then we assign x_{k-1} to y_{j^*-1}. If y_{j^*-1} is already assigned, we add another pair of points (x_{k-2}, y_{j^*-2}) to the conflict chain and repeat the sub-algorithm steps.

Figure 3.1 shows the outline of the algorithm and in figure 3.2 you can find an illustraion of the different cases from the SOPT-article.

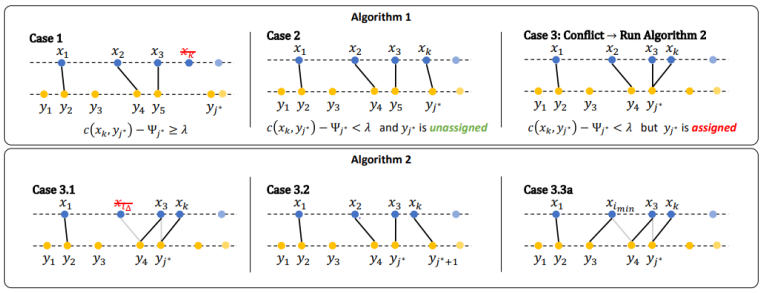


Figure 3.2: Illustration of cases handled by the SOPT algorithm, from [1].

Choice of λ

As already mentioned in 2.14, regularization parameter λ corresponds to the maximum distance mass can be transported, because it influences the impact of the penalty term in the definition of the OPT-Problem defined in 2.4, so it significantly influences the flexibility of mass transport between source and target points. As shown in figure 3.3

(adapted from the SOPT article), varying λ modifies the amount of transportable mass. When λ is relatively low, the algorithm imposes stricter constraints, allowing transport primarily between nearby points, which ensures more localized matching. As λ increases, these constraints relax, allowing the mass to be transported over larger distances. This adjustment leads to a more global alignment of points but also assigns points, that are very far apart, which could be disadvantageous in practical cases. Consequently, selecting an appropriate λ value is essential to balancing local accuracy with global flexibility in partial transport problems.

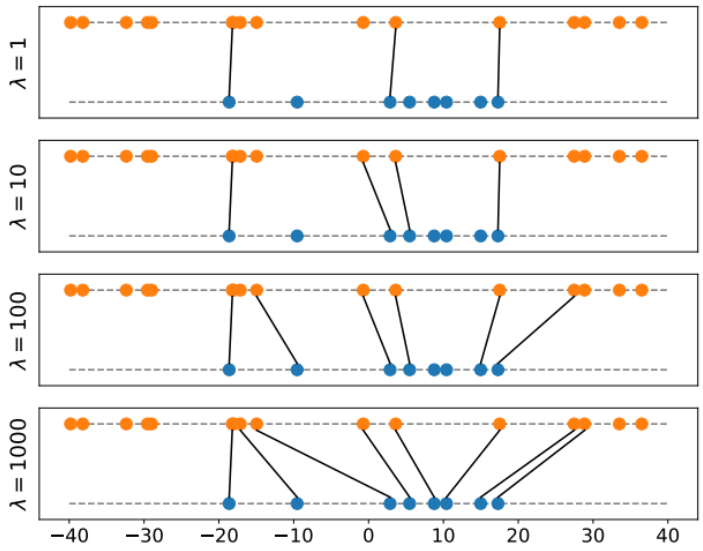


Figure 3.3: Impact of choice of λ, source: [1].

3.2 Slicing and appilcation in Point Cloud Registration

The authors from [1] of the Sliced Optimal Partial Transport (SOPT) algorithm also proposed an application for their 1D-algorithm, designed to align two 3D point clouds by iteratively transforming the source cloud to match the target cloud. This approach uses slicing techniques to simplify the optimization process, where each slice represents a projection of the 3D space onto a 1D line. By iteratively adjusting the scale, rotation, and translation, the source point cloud is aligned with the target cloud.

Given:

- a **source point cloud** X, represented by orange points in the figure 3.5,
- a **target point cloud** Y, represented by green points,

the goal is to compute a transformation T, such that $\hat{Y} := Tx = sRX + \beta$, so we want to obtain the green point cloud Y by iteratively scaling, rotating and shifting the orange one, where s is a scaling factor, R is a rotation matrix and β is a translation vector.
The goal is to achieve an approximate alignment such that $\hat{Y} \approx Y$, using the transformation T.
To solve that problem, the authors used their Algorithm 3.
In figure 3.4, the proposed outline from the SOPT-article [1] for this algorithm is shown.

Algorithm 3: iterative-sopt

Input: $\{x_i\}_{i=1}^n, \{y_j\}_{j=1}^m$, n_0:the # of clean x , N: # of projections
Output: R, s, β

initialize R, s, β, λ, sample $\{\theta_i\}_{i=1}^N \subset \mathbb{S}^2$
for $l = 1, \dots N$ **do**
  $\hat{Y} \leftarrow sRX + \beta$
  Compute transportation plan L of $\mathrm{OPT}_\lambda(\theta_l^T \hat{Y}, \theta_l^T Y)$ by algorithm 1
  $\forall i \in \mathrm{dom}(L)$, $\hat{y}_i \leftarrow \hat{y}_i + (\theta_l^T y_{L[i]} - \theta_l^T \hat{y}_i)\theta$
  Compute R, s, β from $(X[dom(L)], \hat{Y}[dom(L)])$ by ICP (e.g., equations (39)-(42) in [55])
  If $|dom(L)| > n_0$, decrease λ; otherwise, increase λ.

Figure 3.4: Pseudocode of Point Cloud Algorithm, source: [1]

Detailed Steps and Explanation

1. **Initialization**: Initialize the transformation parameters R, s, and β, and select a sample of projection directions $\{\theta_i\}_{i=1}^N \subset S^2$, representing different directions in 3D space along which to slice the point clouds.
2. **Initial Guess**: Compute an initial transformed source point cloud $\hat{Y} = sRX + \beta$ by using the initial parameters above.
3. **Slicing and Optimal Partial Transport**: In this step, we choose one θ_l and slice the 3D point clouds along this direction. The results are two onedimensional measures, on which we can apply Algorithm 1 (SOPT), to compute the optimal transport plan between the onedimensional points of Y and $\hat{Y}$.
4. **Update of** $\hat{Y}$: For each point $\hat{y}_i$ in the transformed source point cloud $\hat{Y}$, move this point in the direction of it's assigned point $y_{L[i]}$. Steps 3 and 4 can be represented using the follwing formula:

$$\hat{y}_i = \hat{y}_i + \left(\theta_l^T y_{L[i]} - \theta_l^T \hat{y}_i\right) \theta_l.$$

In other words, the computed distance between the two onedimensional points $\hat{y}_i$ and $y_{L[i]}$ that were assigned to each other, gets projected back into 3D and gets added to the old point $\hat{y}_i$.

5. **Transformation Update Using ICP**: In the next step, we update our tranformation parameters R, s, and β, such that $\hat{Y} \approx sRx + \beta$ using Iterative Closest Point (ICP) algorithms. It is possible to find more details in [1].

6. **Lambda Adjustment**: If the domain dom(L) exceeds a threshold, increase λ to penalize further deviations. Again, for more details look at the article.

We repeat Steps 3-6 for each θ_i, until we obtain the green point cloud.

In the Chapter after the next, we will try to use this technique for our PET-problem to adapt the formula, so that we can use it in our case.

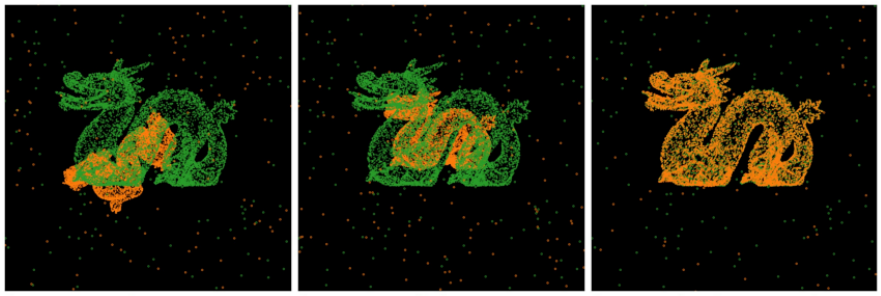


Figure 3.5: Example of iterative slicing for aligning the source (orange) and target (green) point clouds.

4 Physical Background of Positron Emission Tomography

Positron Emission Tomography (PET) is an imaging technique used to visualize metabolic processes inside the body, see for example [3], Chapter 6 or [6]. This method is based on the detection of gamma photons produced by the annihilation of positrons emitted from a radioactive tracer. The tracer is typically a biologically active molecule, such as glucose, labeled with a positron-emitting radionuclide (e.g., Fluor-18). Once the tracer is injected into the patient's body, it distributes according to the body's metabolic activity.

The fundamental physical process behind PET imaging starts with the decay of the radionuclide, which emits a positron (β^+-decay). The positron travels a short distance in the tissue before colliding with an electron. This collision leads to the annihilation of the positron and electron, a process that produces two gamma photons of 511 keV each. Most importantly, these photons are emitted in exactly opposite directions (180 degrees apart) due to the conservation of momentum in the annihilation process.

The patient lies inside a ring-shaped scanner that contains multiple detectors arranged in a circular shape around the body. These detectors have the function of capturing the gamma photons. When two photons are detected almost simultaneously by detectors on opposite sides of the scanner (within a small time window), the system registers this event as a "coincidence". The line connecting the two detectors is referred to as a line of response (LOR), which represents the possible path along which the annihilation occurred. Multiple such lines are detected for each radioactive decay, and over time, this results in a large number of LORs, providing the data for image reconstruction.

There are also a few challenges in PET imaging. One of the main challenges is the scattering of photons, which can occur due to interactions with surrounding tissues before reaching the detector. Scattering changes the direction of the photons, which can lead to errors in determining the exact line of response. Algorithms need to correct those

scattering events during the reconstruction process. For further details on this topic again, look at [3], Chapter 6 or [6].

The primary task of PET imaging is to reconstruct a 3D representation of the tracer distribution inside the body, which indirectly reflects the biological or metabolic processes of interest. Conventional PET systems are optimized for imaging larger structures, such as tumors or organs, and typically operate with relatively high levels of radioactivity.

In our specific application, however, we aim to detect a small number of individual cells, which significantly lowers the amount of radioactivity involved. This leads to increased relative noise. Furthermore, at these lower levels of radioactivity, the effects of photon scattering becomes worse, leading to higher levels of noise.

5 Gradient Descent Algorithm

The **Gradient Descent (GD)** and **Stochastic Gradient Descent (SGD)** are important optimization algorithms widely used in machine learning and data analysis. Both methods aim to minimize a cost function $f(x)$ by iteratively updating the model parameters in the direction that reduces the cost. Below is a detailed description of both methods.

5.1 Gradient Descent (GD)

Gradient Descent is a first-order optimization algorithm that seeks to find the minimum of a multivariate differentiable function $f : \mathbb{R}^d \to \mathbb{R}$. Given a function $f(x)$ defined as the average of n individual components $f_i(x)$, where:

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x),$$

the Gradient Descent method calculates the full gradient $\nabla f(x)$ at each iteration, which involves computing $\nabla f_i(x)$ for every $i = 1, 2, \ldots, n$. The model parameters x are updated iteratively using:

$$x^{k+1} = x^k - \alpha_k \nabla f(x^k),$$

where α_k is the step size or learning rate at iteration k. This method provides an exact update direction based on the entire dataset, which often leads to stable convergence, but computing the gradient $\nabla f(x^k)$ can also be computationally expensive and slow, particularly for large datasets.

5.2 Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) is a variant of the Gradient Descent method that aims to reduce the computational cost by approximating the full gradient with a single

randomly chosen component $f_{i_k}(x)$ at each iteration. In SGD, only one function $f_{i_k}(x)$ is selected at each iteration k, where i_k is an index chosen from the set $\{1, 2, \ldots, n\}$. The update rule for SGD is thus given by:

$$x^{k+1} = x^k - \alpha_k \nabla f_{i_k}(x^k). \tag{5.1}$$

The selection of i_k can follow different strategies, for example one can choose i_k to cycle through $\{1, 2, \ldots, n\}$ in order (Cyclic Selection), or i_k is chosen randomly from $\{1, 2, \ldots, n\}$ at each iteration (Random Selection).
For more details, see [8].

6 The Algorithm

Algorithm Idea: Cell Localization Using PET and SOPT Minimization

As described earlier, in PET, we want to locate radioactively labeled cells that have been injected into a patient. The measurements consist of lines where a cell must lie on. However, not all of these measurement lines are accurate, as some may contain noise or false signals. Typically, each cell generates multiple measurement lines, let's say with an average of $\mathbf{c}$ lines per cell.
The objective is to develop an algorithm that can localize the cells using the data. The core idea of the algorithm is as follows:

1. **Initialization:**
    - For each measurement line, place a point on that line. These points will form a configuration X.
    - Additionally, generate a set of points randomly distributed within $\mathbb{R}^3$ to form a configuration Y. In this configuration, each set of $\mathbf{c}$ points should overlap at a single location, effectively forming a "point of mass $\mathbf{c}$" that represents a potential cell.
2. **Optimization Process:**
    - The goal is to simultaneously adjust the positions of the points in configurations X and Y such that their **SOPT distance** is minimized.
    - The points in X can only move along the measurement lines they were initially placed on. This constraint reflects the fact that the true cell location must lie somewhere along these lines.
    - The points in Y, however, must move as clusters of $\mathbf{c}$ overlapping points. This constraint ensures that each cluster represents a single cell, as each cell should generate approximately $\mathbf{c}$ measurement lines.

By iteratively minimizing the SOPT distance while keeping the moving constraints in mind, the algorithm wants to align the clusters in Y with the points in X and therefore find the positions of the cells.
Let m be the number of cells we are looking for and n be the number of measurements, where n is larger than m, because we have multiple measurement lines per cell.
We beginn with initializing our points $X^0 = \{x_i^0\}_{i=1}^n$ and $Y^0 = \{y_j^0\}_{j=1}^m$.
In every iteration k, we have the constraint, that every point x_i^k has to lie on it's correspoding line of response LOR_i, so $x_i^k = p_i + s_i^k v_i$, where $p_i, v_i \in R^3$ define the line LOR_i
From that, we derive the measures μ and ν as follows:

- Points X^k: $\mu = \sum_{i=1}^n \delta_{x_i^k}$, where $x_i^k = p_i + s_i^k v_i$ for every iteration k and every index i,
- Points Y^k with mass $\mathbf{c}$: $\nu = \sum_{j=1}^m c\delta_{y_j^k}$, where $\mathbf{c} = \frac{n}{m}$.

Here, of course, we assume that $\mathbf{c}$ is an integer.
Note, that it is possible to view the points with mass $\mathbf{c}$ as a number of $\mathbf{c}$ points with mass 1. We will need that to use the SOPT-algorithm to assign the points X and Y to each other. As a result we get that one point with mass $\mathbf{c}$ (which represents one cell) can be assigned to up to $\mathbf{c}$ points in X. We need that because the SOPT-algorithm can only handle points with mass 1.
After we solved the SOPT problem, it is our objective to minimize the Wasserstein-2-distance between those points in X and Y, that where assigned to each other, while paying attention to the constraints that μ must fit to the measurements, so the points X can only move along their lines of responses.

6.1 Outline of algorithm/intuition/informal description

In this section, the informal description of the steps in the cell locating algorithm and it's pseudocode are provided.

- **Step 0 (Initilization)**:
  Given $\{LOR_i\}_{i=1}^n$, initialize $X^0 = \{x_i^0\}_{i=1}^n$ and $Y^0 = \{y_j^0\}_{j=1}^m$ and choose λ and $\{\theta_i\}_{i=1}^N \subset S^2$.
  Then, in the k-th iteration, we do the following:

- **Step 1 (Slicing and solving the OPT-problem in 1D)**:

  In the first step, we choose one θ^k and project both sets of points X^k and Y^k into 1D by multiplying with θ^k. Then the SOPT-Algorithm assigns those points to each other, depending on their 1D-distance. As a result, we get a vector $L : \{1, \dots, n\} \to \{1, \dots, m\}$, where $L[i] = j$ describes, that the 1D-point $x_i^k \theta^k$ was assigned to the point $y_j^k \theta^k$.
  Note that by viewing the points in Y as points with a mass (i.e. $\mathbf{c}$ overlapping points), it is now possible to use the SOPT-algorithm for points with masses and therefore we can assign multiple points in X to one point with mass $\mathbf{c}$ in Y.

- **Step 2 (Move points Y towards the middle of their assigned points X)**:
  Because it is possible and also likely that one point with mass $\mathbf{c}$ in Y was assigned to multiple points in X, the next step is calculating the new points in Y^{k+1} by computing the mean of their assigned points in X, so compute:

$$y_j^k = \frac{1}{m_j} \sum_{l \in L^{-1}(j)} x_l^k, \text{ where } m_j = |L^{-1}(j)|.$$

- **Step 3 (Move the points X)**:
  Calculate the 1D-distances of a 1D-Point x_i^k and it's corresponding 1D-point $y_{L[i]}^k$ and project the distance back to 3D. Then add this 3D-vector to the original 3D-point x_i^k:

$$\tilde{x}_i^{k+1} = x_i^k + (\theta^k \cdot y_{L[i]} - \theta^k \cdot x_i)\theta^k$$

- **Step 4 (Orthogonal projection)**:
  Make sure, that the points in X^{k+1}stay on their lines, so project them orthogonally onto their measurement lines, so

$$x_i^{k+1} = P_{LOR_i}(\tilde{x}_i^{k+1}).$$

As an alternative to step 4, we can parametrize the points in X by writing $x_j = p_j + s_j v_j$ and then optimizing s_j instead of x_j, which is what we will do and is described in the next section.
Here, you find the pseudocode for those steps:

Algorithm 1 cell localization

Input: $\{LOR_i\}_{i=1}^n$, N: # of projections, $\lambda \in \mathbb{R}$

initialize $X^0 = \{x_i^0\}_{i=1}^n$ and $Y^0 = \{y_j^0\}_{j=1}^m$, λ, sample $\{\theta_i\}_{i=1}^N \subset S^2$
for $k = 1, \dots, N$ **do**
    Compute transportation plan L of $\mathrm{OPT}_\lambda(\theta_k^T X, \theta_k^T Y)$ with OPT-algorithm
    for all $j = 1, \dots, m$ **do**
        $m_j = |L^{-1}(j)|$
        Compute $y_j = \frac{1}{m_j} \sum_{l \in L^{-1}(j)} x_l$
    end for
    for all $i \in \mathrm{dom}(L)$ **do**
        $x_i \leftarrow P_{LOR_i}\left(x_i + (\theta^k \cdot y_{L[i]} - \theta^k \cdot x_i)\theta^k\right)$
    end for
end for

6.2 Local Convergence Proof

In this section, we provide a local convergence proof. Starting near the ground truth allows us to assume that the point assignments between the sets X and Y do not change after a few iterations. The goal of our local convergence proof is to minimize the Sliced Wasserstein-2 distance between X and Y.

We begin by assuming, that in iteration k, the following set of indices are assigned to the index of the point y_j^k:
$L^{-1}(j) = \{i \in \{1, \dots, n\} \mid L(i) = j\} = \{i \in \{1, \dots, n\} \mid x_i^k \text{ is assigned to } y_j^k\}$
and we try to minimize the following function:

$$f(\{x^k\}_{k=1}^N) := \frac{1}{N} \sum_{k=1}^{N} f_k(x^k),$$

where

$$f_k(x^k) := \frac{1}{2} \sum_{i \in L^{-1}(j)} \Bigg| \underbrace{\langle y_j^k, \theta^k \rangle - \left\langle x_i^k, \theta^k \right\rangle}_{:= f_{k_i}} \Bigg|^2.$$

In the following, we will use the stochastic gradient method to minimize this function. We will use in every iteration k only one part of the sum, so we will only compute the gradient of $f_k(x^k)$.

First, we take the sum over every y_l and include that every y_l is the average of its assigned points in X, so use this formula $y_j^k = \frac{1}{m_j}\sum_{l\in L^{-1}(j)} x_l^k$ to compute

$$f_k(x^k) = \sum_{j=1}^{m} \frac{1}{2} \sum_{i\in L^{-1}(j)} \left| \langle \frac{1}{m_j} \sum_{L^{-1}(j)} x_l^k, \theta^k \rangle - \left\langle x_i^k, \theta^k \right\rangle \right|^2 .$$

By using the parametrization formula $x_i^k = p_i + s_i^k v_i$, we get the following function to minimize:

$$f_k(s^k) = \sum_{j=1}^{m} \frac{1}{2} \sum_{i\in L^{-1}(j)} \left| \langle \frac{1}{m_j} \sum_{l\in L^{-1}(j)} p_l + s_l^k v_l, \theta^k \rangle - \left\langle p_i + s_i^k v_i, \theta^k \right\rangle \right|^2 , \tag{6.1}$$

where now we minimize subject to s^k instead of x^k. Doing that for every iteration k, we get the sliced Wasserstein-2-distance:

$$f(\{s^k\}_{k=1}^N) = \frac{1}{N}\sum_{k=1}^{N}\sum_{j=1}^{m} \frac{1}{2} \sum_{i\in L^{-1}(j)} \left| \langle \frac{1}{m_j} \sum_{l\in L^{-1}(j)} p_l + s_l^k v_l, \theta^k \rangle - \left\langle p_i + s_i^k v_i, \theta^k \right\rangle \right|^2 .$$

We want to minimize the function f. To do this, we want to use the stochastic gradient descent algorithm (5.1) with $i_k = k$, so in every iteration k we update all the points $\{s_i^k\}_{i=1}^n$ using the gradient of the function $f_k(s^k)$ defined in equation 6.1.
We choose one index $i_t \in 1, ..., n$ and denote $j_t = L[i_t]$ as the index i_t was assigned to. Then the stochastic gradient formula looks like this (for the k-th iteration and the i_t-th point x_{i_t} or s_{i_t}) : $\mathrm{s}_{i_t}^{k+1} = s_{i_t}^k - \alpha_k \nabla_{s_{i_t}^k} f_k(s^k)$ Where we denote $\nabla_{i_t} = \nabla_{s_{i_t}^k}$. Now we need to calculate the gradient:

$$\nabla_{i_t} f_k(s^k) = \nabla_{i_t}\Big(\sum_{j=1}^{m} \frac{1}{2} \sum_{i\in L^{-1}(j)} \underbrace{\left| \langle \frac{1}{m_j} \sum_{l\in L^{-1}(j)} p_l + s_l^k v_l, \theta^k \rangle - \left\langle p_i + s_i^k v_i, \theta^k \right\rangle \right|^2}_{=f_{k_i}} \Big).$$

The first sum disappears, because every point x_i^k can only be assigned to one y_j^k, so only one of the summands stays when we take the derivative. Then, we can calculate:

$$\nabla_{i_t} f_{k_i}(s_i^k) = \nabla_{i_t}\left(\langle \frac{1}{m_{j_t}} \sum_{l \in L^{-1}(j)} p_l + s_l^k v_l, \theta^k \rangle\right) \quad - \quad \nabla_{i_t}\left(\left\langle p_i + s_i^k v_i, \theta^k \right\rangle\right)$$

$$= \quad \frac{1}{m_{j_t}} \left\langle v_{i_t}, \theta^k \right\rangle \quad - \quad \begin{cases} \langle v_i^k, \theta^k \rangle & \text{if } i = i_t \\ 0 & \text{if } i \neq i_t. \end{cases}$$

Next, we calculate

$$\begin{aligned} \nabla_{i_t}(f_{k_i}(s_i^k))^2 &= 2\, f_{k_i}(s_i^k)\ \nabla_{i_t} f_{k_i}(s_i^k) \\ &= 2\left(\langle y_j^k, \theta^k \rangle - \left\langle x_i^k, \theta^k \right\rangle\right)\left(\frac{1}{m_j}\left\langle v_{i_t}, \theta^k \right\rangle - \begin{cases} \langle v_i, \theta^k \rangle & \text{if } i = i_t \\ 0 & \text{if } i \neq i_t. \end{cases}\right) \end{aligned}$$

Now, to compute the final gradient by taking the sums and noting that only one indice j_t stays, which the i_t-th indice was assigned to, explicitly this is the indice $j_t = L[i_t]$:

$$\nabla_{i_t} f_k(s^k) \tag{6.2}$$

$$= \sum_{j=1}^{m} \frac{1}{2} \sum_{i \in L^{-1}(j)} \nabla_{i_t}(f_{k_i}(s_i^k))^2 \tag{6.3}$$

$$= -\left(\langle y_{j_t}^k, \theta^k \rangle - \left\langle x_{i_t}^k, \theta^k \right\rangle\right)\langle v_{i_t}, \theta^k \rangle + \frac{1}{m_{j_t}}\left\langle v_{i_t}, \theta^k \right\rangle \sum_{j=1}^{m} \sum_{i \in L^{-1}(j)} \left(\langle y_j^k, \theta^k \rangle - \left\langle x_i^k, \theta^k \right\rangle\right), \tag{6.4}$$

where the first part is exactly the formula we wanted if we plug in the gradient into formula 6.2.

We derived our formula by using stochastic gradient descent, which converges, so our algorithm doas also (locally) converge.

As I already mentioned above, the parametrization of the points X, to make sure they stay on their lines (and therefore updating the scalars s), is equivalent to updating the points x and then orthogonally project the points x onto their lines. We can see that by just calculating the orthogonal projection P_{LOR_i} onto line LOR_i

$$\begin{aligned} x_{i_t}^{k+1} &= P_{LOR_i}\left(x_{i_t}^k - \alpha_k \nabla_{x_{i_t}^k} f_k(x^k)\right) \\ &= x_{i_t}^k - P_{LOR_i}\left(\alpha_k \left(\langle y_{j_t}^k, \theta^k \rangle - \left\langle x_{i_t}^k, \theta^k \right\rangle\right)\theta^k\right) \\ &= x_{i_t}^k - \alpha_k \left(\langle y_{j_t}^k, \theta^k \rangle - \left\langle x_{i_t}^k, \theta^k \right\rangle\right)\theta^k \frac{v_{i_t}}{|v_{i_t}|}\frac{v_{i_t}}{|v_{i_t}|}. \end{aligned}$$

If we look at the first part of formula 6.4 and plug this into $x_{i_t}^{k+1} = p_{i_t} + s_{i_t}^{k+1} v_{i_t}$, we get that

$$\begin{aligned} x_{i_t}^{k+1} = p_{i_t} + s_{i_t}^{k+1} v_{i_t} & \\ &\overset{(6.2)}{=} p_i + \left(s_{i_t}^k - \alpha_k \nabla_{i_t} f_k(s^k) \right) v_i \\ &\overset{(6.4)}{=} p_i + \left(s_{i_t}^k + \alpha_k \left(\langle y_{j_t}^k, \theta^k \rangle - \left\langle x_{i_t}^k, \theta^k \right\rangle \right) \langle v_{i_t}, \theta^k \rangle \right) v_{i_t} \\ &\overset{x_{i_t}^k = p_{i_t} + s_{i_t}^k v_{i_t}}{=} x_i^k + \alpha_k \left(\langle y_{j_t}^k, \theta^k \rangle - \left\langle x_{i_t}^k, \theta^k \right\rangle \right) \theta^k v_{i_t} v_{i_t}, \end{aligned}$$

which is the same as calculating the orthogonal projection above, if the direction vectors v_i are normed for every line.

6.3 Initialization

To make sure we can use the local convergence proof, we have to make sure that we start near our ground truth, so that the assignments of our OPT-algorithm do not change so much.
We can look at this from an applied point of view: Often, the scanners, in which the patient lies in, is very big, while the cells are very close to each other. If we choose a random initialization of the points and place the points X and Y randomly in the big scanner, the algorithm will not converge.
For that reasons I used a discrete version of FBP.

Points Y

As we know from Inverse Problems, in medical imaging techniques, the radon transform of the image gives the measurement. So for the initialization, the idea is to use a (rough) discrete inversion of the radon transform. Let's denote by LOR the vector, which contains in the i-th component the start vector p_i and the direction vector v_i, from which we can derive the Line LOR_i on which the point x_i lies. Then we can compute the discrete backprojection on one point $z = (z_1, z_2, z_3) \in \mathbb{R}^3$ by the following formula:

$$BP(z_1, z_2, z_3) = \sum_{i=1}^{n} G(d((z_1, z_2, z_3), LOR_i)),$$

where $d((z_1, z_2, z_3), LOR_i)$ is the smallest distance between the point z and the line LOR_i and G is a Gaussian filter, defined as $G(d) := \exp(-\frac{d^2}{2\sigma^2})$. The variance σ^2 influences how wide the filter is, thus a bigger σ leads to stronger smoothness.
In this formula, we thicken up the measurement lines with the Gaussian filter. $G(d)$ corresponds to how much a point contributes to the backprojection, depending on the distance the point has to the line.
As mentioned above, we only do a rough initialization and calculate the backprojection on equidistant points. Then, depending on how many cells we are looking for, we take the maximum (for example the top $m * 3$) values of the discrete backprojection. The points, where the values of the BP-function are maximal, are then our candidates for the initial points Y.
After that, I used the k-means cluster algorithm to cluster those maximum points to get exactly m initial points.

Points X

After I initialized our points Y, we get our points X by choosing for every point in X one Point in Y randomly and project it orthogonally onto the line of response of the point in X. This is random but still makes sure that the initial points are near the ground truth.

6.4 Choice of parameters

The algorithm uses two key parameters, λ and `max_distance`, which control the assignment of points from set X to set Y. Both parameters are chosen so that the transport cost based on distance is small while trying to deal with scattering. However, they operate on different principles and affect the matching process in different ways.

The Role of λ

As already mentioned before, the parameter λ from the OPT-algorithm from [1] determines the threshold for the **1D distance** between the projected points from sets X and Y. It controls the initial assignment by specifying how close the projections of two points must be for them to be considered a match. If the absolute 1D distance between a point x_i and a point y_j exceeds λ, the assignment is removed (i.e., $L[i] = -1$).
Bigger λ values allow for a higher distance when we assign points, meaning that a greater number of points from set X may be assigned to points in set Y. However, this

increased flexibility can lead to less precise assignments, as points that are relatively far apart in 3D space may still be considered close enough in 1D.
Smaller λ values, on the other hand, lead to fewer assignments. This results in a more precise, but possibly sparse, matching. There could be many points that remain unassigned, which could lead to less moving of points in the right direction.

An additional aspect of the parameter λ is its influence on the algorithm's sensitivity to noise and scattering.
Smaller λ values lead to a stricter criterion for matching points, because we filter assignments that could be influenced by noise or scatter lines. This is because only points that are very close to each other in the 1D projection are considered for matching. Irrelevant or noisy data are less likely to be assigned and therefore the impact of noise is reduced in the solution.
On the other hand, **larger λ values** allow more assignments to occur, which increases the likelihood that some of these matches might be influenced by noise. This can result in the algorithm erroneously assigning points that are very far apart from each other in the original 3D space.
Thus, by decreasing λ, we assume that we can control the algorithms robustness against noise, especially in scenarios where there is a lot of scattering, as it is in ours.

The Role of `max_distance`

While λ handles the initial assignment based on the 1D projection, the `max_distance` parameter evaluates the assignments made earlier using the **3D distance**. The parameter λ only takes 1D-distances into account, which can get problematic when the 1D-distance does not represent the actual 3D-distance of two points, for example can the 1D-distance be very low, leading to an assignment in the OPT-algorithm, while the 3D-distance is very large. Therefore I introduced the parameter `max_distance`. Assuming a group of x_i points is assigned to a single y_j point, the average 3D distance between the assigned x_i points and the corresponding y_j point is calculated:

$$d_{\text{avg}} = \frac{1}{n_j} \sum_{i \in A_j} \|x_i - y_j\|_2,$$

where $A_j = \{i \mid L[i] = j\}$ is the set of indices assigned to y_j, and n_j is the number of assigned points. If any point x_i has a 3D distance to y_j that exceeds `max_distance`$\times d_{\text{avg}}$, the assignment is removed. In our examples, the parameter `max_distance` had a value

between 1 und 3, to give an idea in which range the parameter could be chosen.

Interplay between λ and `max_distance` during the Algorithm

The parameter `max_distance` is dynamic, because its influence depends on the configuration of the points at each iteration. As the algorithm progresses, points from sets X and Y are iteratively adjusted, which hopefully results in decreasing distances between assigned points. This reduction in distances leads to a decrease in the value of d_{avg}. Thus, over the course of the algorithm, assignments become increasingly stringent, allowing only those points that are very close in 3D space to remain matched.
In contrast, at least in our case, λ remains constant throughout the algorithm's execution, setting a fixed boundary for the initial assignment of the OPT-algorithm based on the 1D projection. It influences primarily the early stages of the matching process, ensuring that only points that are relatively close in the projected 1D space are considered.

Conclusion: Comparing the impact of the Parameters

In summary, while both parameters λ and `max_distance` try to ensure optimal matching, they operate on different principles and impact different stages of the assignment process:

- λ controls the breadth of the initial assignments based on a 1D projection, acting as a static threshold throughout the algorithm.
- `max_distance` refines the initial assignments by enforcing a dynamic 3D proximity constraint that becomes stricter as the points converge.

We hope that the interplay between those parameters leads to the best possible assignments and therefore to an optimal solution of the algorithm.

7 Numerical Results

7.1 Example

In this chapter, I show the numeric results of the algorithm.
You can see the basic setting in figure 7.1, where the scanner is a cylinder with radius 447.6 mm ($x-y$-plane) and height -114 mm$<= z <= 114$ mm. We are looking for five cells, which are represented by green points and which are relatively close to each other, i.e. have a distance of approximately 10mm. Shown are only 30 lines of responses for the second cell for better clarity. In this example, we have 300 lines, 5 cells, scattering of 30-40% and used 600 iterations for the final result seen in figure 7.4.
In figures 7.2 and 7.3 you see the initialization as described in Chapter 6.3. Again, the green points represent the cells, the red dots represent the top 25 cells, which are the result of the discrete filtered backprojection. The blue crosses are the final initial Y-points we use, which we get by using the k-means cluster algorithm applied to our red points. The black dots in figure 7.3 are the initial X-points, again only shown for 30 measurement lines.

Finally, in 7.5, you see how the algorithm converges over the course of 600 iterations. The red dots represent our guesses for the positions of the cells, while the green dots are the ground-truth positions and the blue dots represent our X-points.
The execution of this example with 600 iterations and 300 measurements took approximately one minute, whereas on the university server, it required about half the time.

Also note that the error calculation is based on the Euclidean distance, also known as the 2-norm.

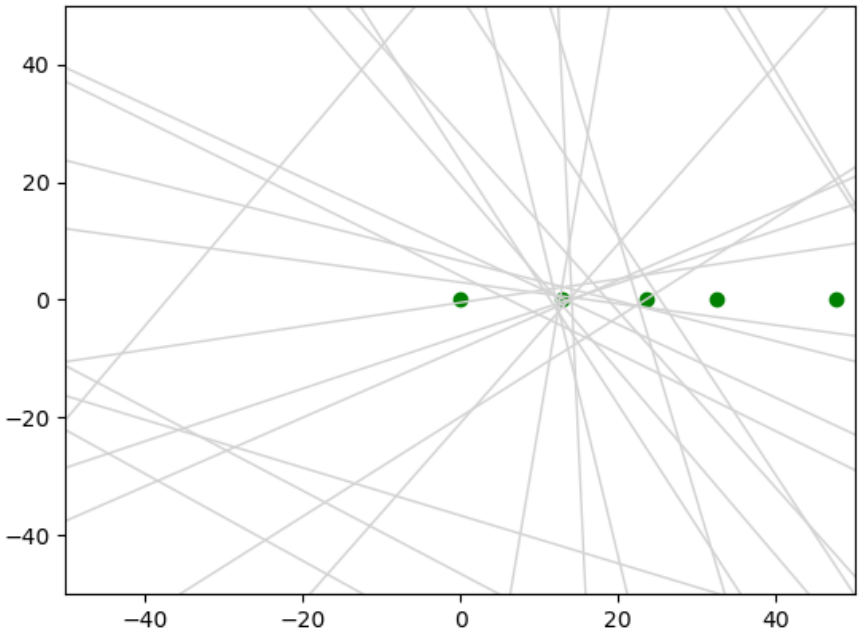


Figure 7.1: Basic setting: 5 cells (green) and 30 measurement lines for the second cell (grey) including some scatter lines

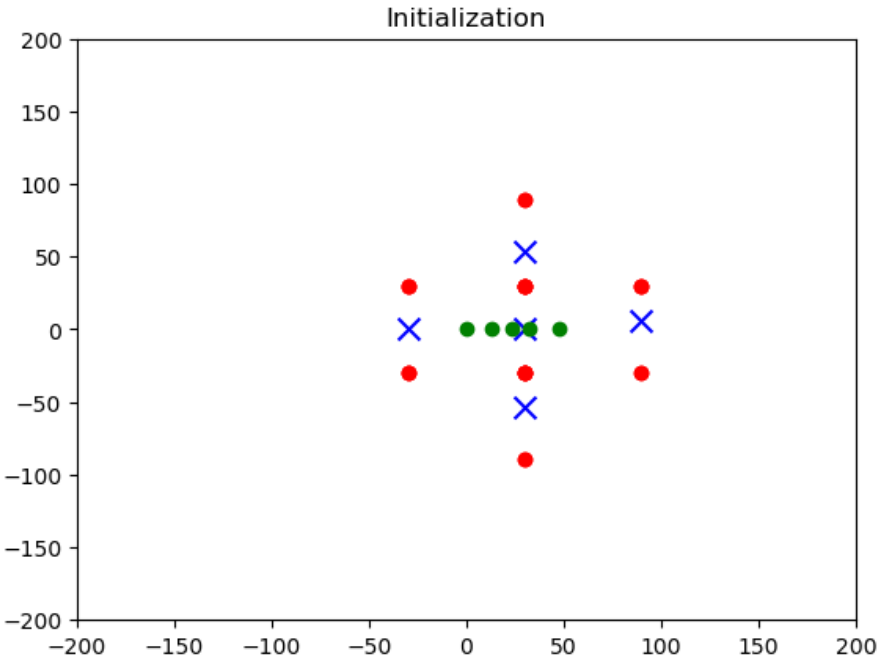


Figure 7.2: Initialization of Y-points: Result of the top 25 values of the discrete FBP (red) and final initial Y-points with k-means (blue)

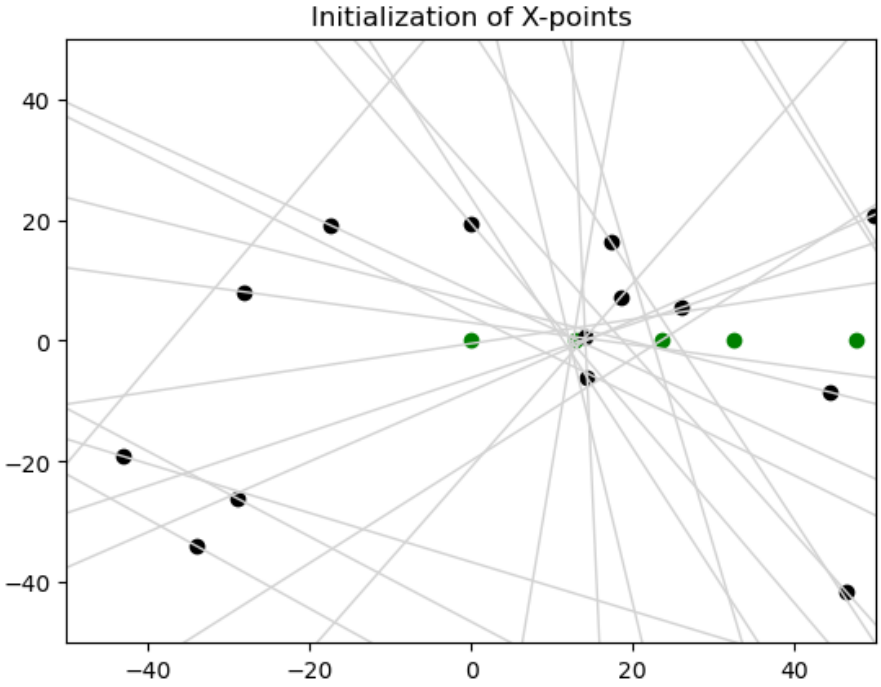


Figure 7.3: Initialization of X-points (black) for 30 measurement lines

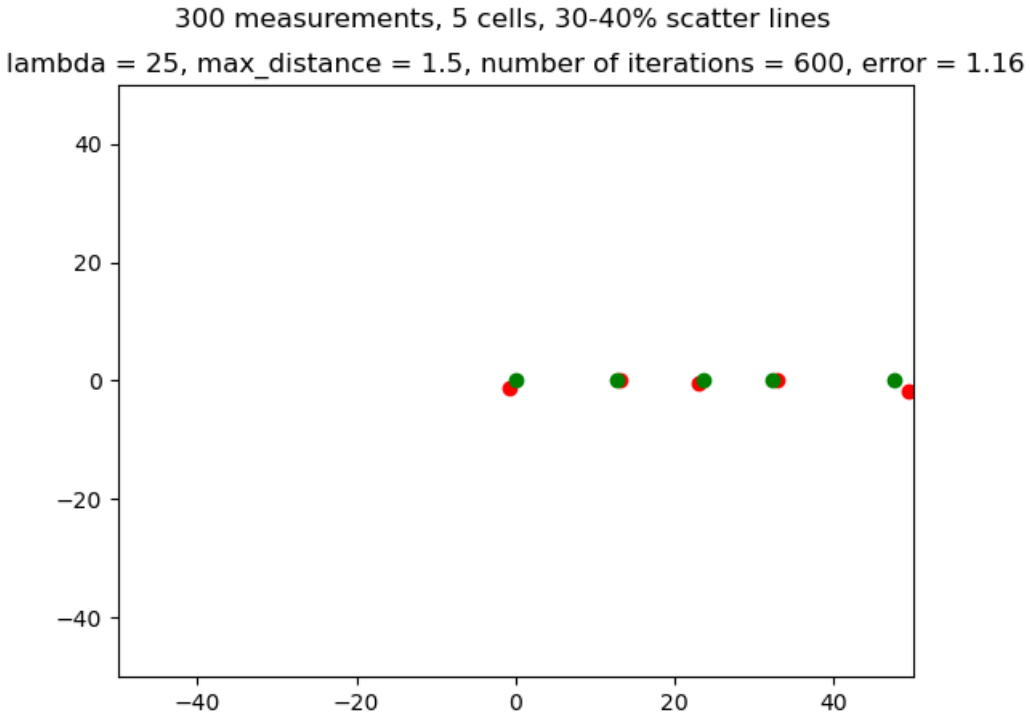
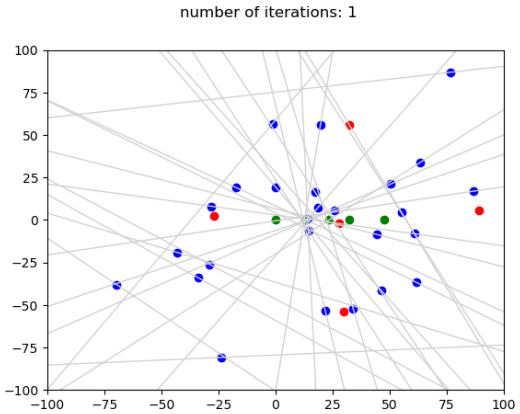
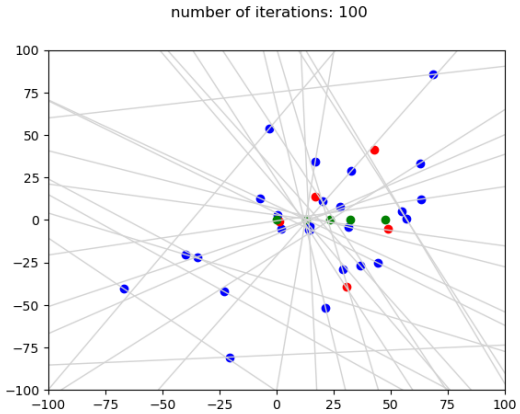


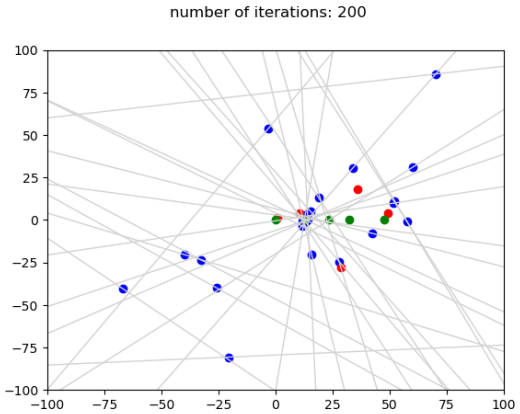
Figure 7.4: Result of the algorithm for the given parameters (red) and ground truth (green)



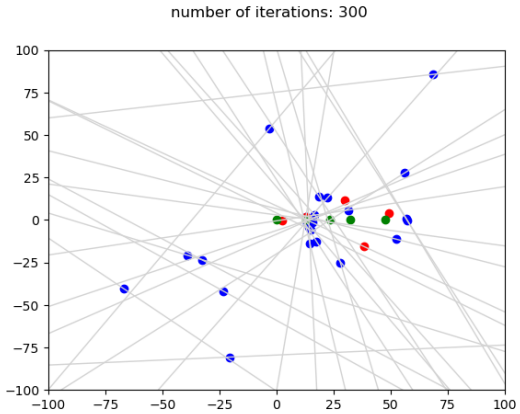
(a) 1 iterations



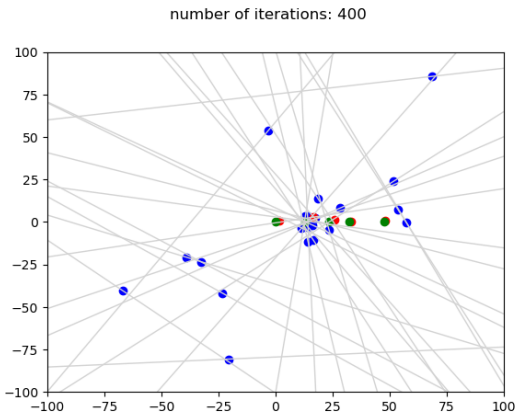
(b) 100 iterations



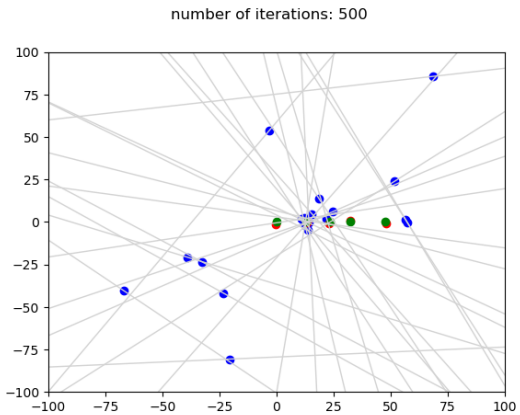
(c) 200 iterations



(d) 300 iterations



(e) 400 iterations



(f) 500 iterations

Figure 7.5: Convergence of the algorithm over the course of 500 iterations

7.2 Parameter Analysis

In this section, I tested combinations of `max_distance` and λ to determine how these parameters should be selected depending on the proportion of noise in the measurements and visualized those results in 7.7.
The noise ratios were tested in the range of 0.1 to 1.1. The `max_distance` values tested were in the range of 1 to 6 and the λ values tested between 10 and 900.
Ten different examples were generated, each containing five cells positioned differently. The distance between each adjacent cell was about 10 mm.
After the initialization step, the error per cell was in the range of 30-40 mm and a cell is considered detected if its final distance is less than 2 mm.
Each test case consisted of 300 correct measurements, with additional noise added based on the noise ratio. For instance, at 50% noise, a total of 450 measurements were used.

For each parameter combination and example, the algorithm was run for 600 iterations and the backprojection was evaluated with standard deviation $\sigma = 20$ at 16^3 different points.

An analysis was performed on all parameter combinations. The results can be seen in figures 7.7, where I visualized all parameter combinations as dots, where on average, more than 97% of the cells were detected. The following conclusions can be drawn from those pictures:
For low noise ratios, the choice of λ is not critical; a wide range of values can still produce good results. However, `max_distance` should not be too large and performs best when set between 1 and 2.5.

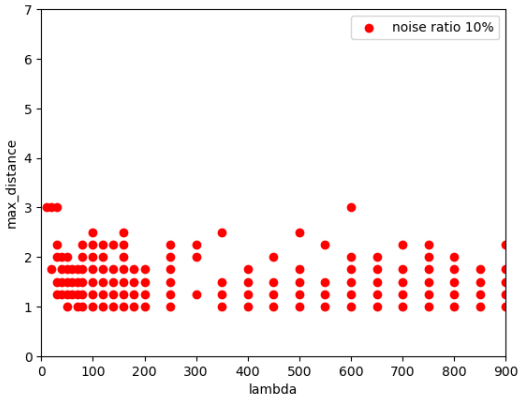
As the proportion of noise increases, selecting an appropriate λ becomes crucial. Larger values of λ allow noise to have a greater influence on the result. For that reason, as noise increases, a smaller λ produces better result. We can also see, that a larger `max_distance` becomes beneficial, likely because if we choose both parameters too small, both parameters limit the amount of points assigned to each other. If λ is chosen smaller while `max_distance` is also very small, fewer points are assigned to each other and if we have the same number of iterations, fewer cells are moved in the right direction at each iteration.

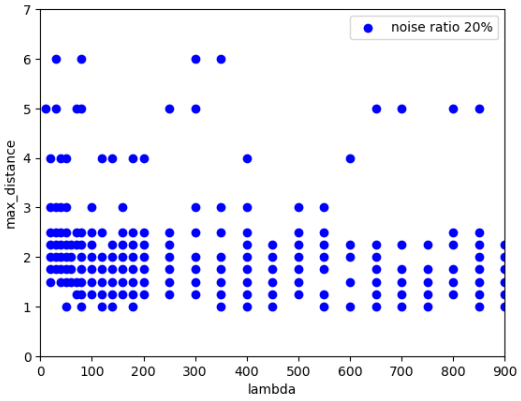
When noise is very high, fewer parameter combinations achieve satisfactory results. In

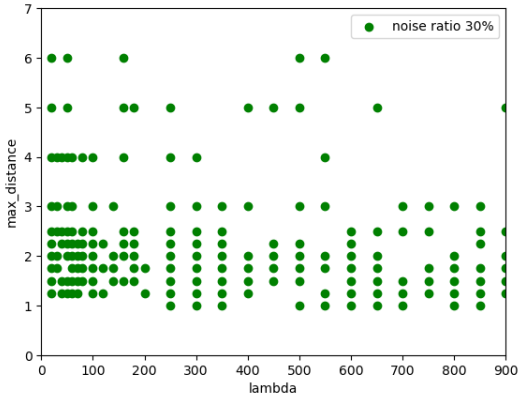
this case it is very important to choose λ very small.

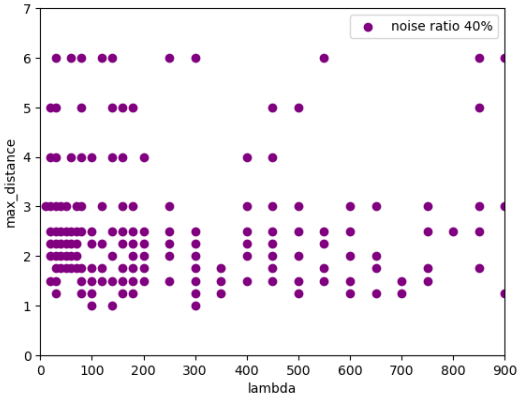
We can assume that the choice of λ strongly depends on the distance between cells. Further investigation with additional test cases, where the cells have different distances to each other, is necessary to validate this observation. However, such an exploration was not feasible within the time constraints of this study.

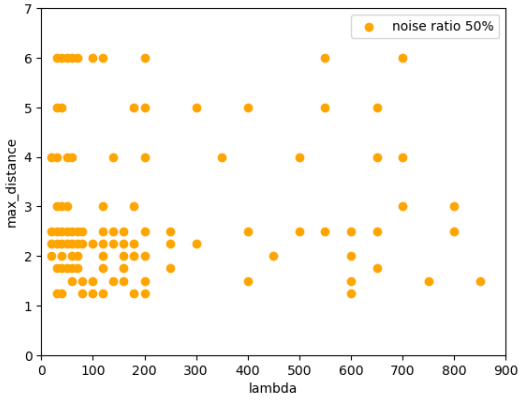
The findings indicate that parameter tuning is highly dependent on the noise ratio in the measurement data. While smaller values of `max_distance` are beneficial for low noise scenarios, larger values become advantageous as noise increases. The parameter λ should be carefully chosen to balance noise suppression and flexibility in adjustments. Further experiments with diverse cell distances could provide further insight into optimizing these parameters effectively.

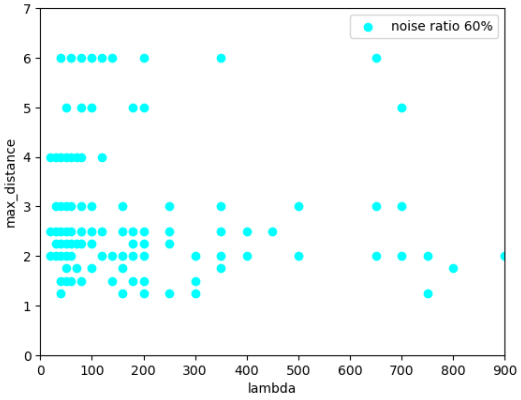

noise ratio 10%
max_distance
lambda


noise ratio 20%
max_distance
lambda


noise ratio 30%
max_distance
lambda


noise ratio 40%
max_distance
lambda


noise ratio 50%
max_distance
lambda


noise ratio 60%
max_distance
lambda

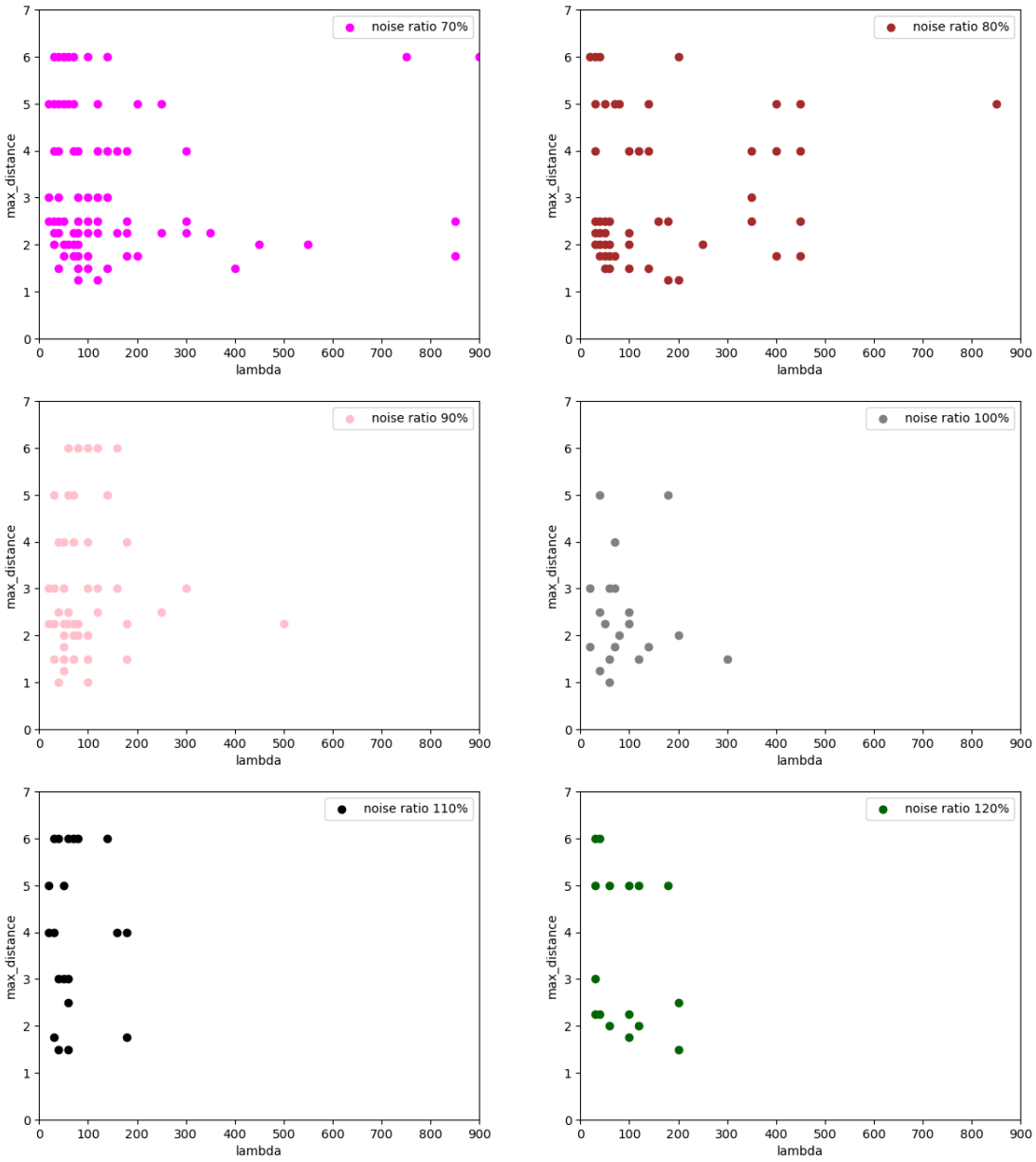


Figure 7.7: Combination of parameters, that achieved an accuracy of 97% or more. The `max_distance` values tested were $[1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 3, 4, 5, 6]$, and the λ values tested were $[10, 20, 31, 40, 50, 60, 70, 80, 100, 120, 140, 160, 180, 200, 250, 300, 350, 400, 450, 500, 550, 600, 650, 700, 750, 800, 850, 900]$

8 Conclusion and Outlook

In this thesis, we presented an approach to cell localization in Positron Emission Tomography (PET) using the Sliced Optimal Partial Transport (SOPT) algorithm. Using optimal transport theory and slicing techniques, our method approximates cell positions on the basis of PET measurement data. The algorithm showed reasonable performance under different noise levels when choosing the parameters accordingly, which highlights its potential for further exploration.

The numerical experiments confirmed that the choice of algorithm parameters `max_distance` and λ, significantly impacts the accuracy of the algorithm. For low noise scenarios, a wide range of parameter values yielded good results, whereas higher noise levels required more careful tuning to achieve the same results. The findings suggest that a dynamic adjustment of these parameters could improve the algorithm's results. However, further parameter analyses would be necessary but are too time-consuming within the scope of this work.

Despite the promising aspects, several challenges remain. The current approach was primarily tested with uniform cell distributions, although other distributions could also be applicable. Different point configurations were not explored in detail, but it would be interesting to see how they influence the choice of parameters, that has to be made to achieve good results. Future work could focus on extending the algorithm to handle moving cells, which would be a natural extension of the existing framework.
Additionally, further research is required to improve computational efficiency if the algorithm is used on larger datasets. Optimizing the implementation and exploring parallel computing techniques could reduce processing time and make the approach more practical.

In conclusion, the proposed algorithm provides a starting point for a different approach to PET cell localization and hopefully contributes to the broader exploration of the algorithms in medical imaging.

References

[1] Yikun Bai et al. *Sliced Optimal Partial Transport*. 2023. arXiv: 2212.08049 [cs.LG]. URL: https://arxiv.org/abs/2212.08049.

[2] Mathias Beiglböck, Pierre Henry-Labordère, and Friedrich Penkner. *Model-independent Bounds for Option Prices: A Mass Transport Approach*. 2013. arXiv: 1106.5929 [q-fin.PR]. URL: https://arxiv.org/abs/1106.5929.

[3] National Research Council. *Mathematics and physics of emerging biomedical imaging*. National Academy Press, Washington, DC (United States), Dec. 1996. URL: https://www.osti.gov/biblio/510447.

[4] Matthias Liero, Alexander Mielke, and Giuseppe Savaré. "Optimal Entropy-Transport problems and a new Hellinger–Kantorovich distance between positive measures". In: *Inventiones mathematicae* 211.3 (Dec. 2017), 969–1117. ISSN: 1432-1297. DOI: 10.1007/s00222-017-0759-8. URL: http://dx.doi.org/10.1007/s00222-017-0759-8.

[5] Gabriel Peyré and Marco Cuturi. *Computational Optimal Transport*. 2020. arXiv: 1803.00567 [stat.ML]. URL: https://arxiv.org/abs/1803.00567.

[6] D Townsend. "Physical Principles and Technology of Clinical PET Imaging". In: *Annals of the Academy of Medicine, Singapore* 33 (Apr. 2004), pp. 133–45. DOI: 10.47102/annals-acadmedsg.V33N2p133.

[7] Cédric Villani. "The Wasserstein distances". In: *Optimal Transport: Old and New*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 93–111. ISBN: 978-3-540-71050-9. DOI: 10.1007/978-3-540-71050-9_6. URL: https://doi.org/10.1007/978-3-540-71050-9_6.

[8] Stephen J. Wright and Benjamin Recht. *Optimization for Data Analysis*. Cambridge University Press, 2022.

Declaration of Academic Integrity

I hereby confirm that this thesis, entitled _____

_____,

is solely my own work and that I have used no sources or aids other than the ones stated. All passages in my thesis for which other sources, including electronic media, have been used, be it direct quotes or content references, have been acknowledged as such and the sources cited. I am aware that plagiarism is considered an act of deception which can result in sanction in accordance with the examination regulations.

I confirm that I am aware that my work may be cross-checked with other texts to identify possible similarities and that it may be stored in a database for this purpose.

I confirm that I have not submitted the following thesis in part or whole as an examination paper before.

_____

(date, signature of student)