哈爾濱工業大學
Harbin Institute of Technology

机器人技术与系统国家重点实验室
State Key Laboratory Of Robotics and System

Scan Matching in 2D SLAM

张明明
robotsming@gmail.com
2016年7月8日

0.背景：雷达-单目融合

1.闭环检测。

2.跟踪失败互补。（空旷，纹理少）

KITTI dataset
（单目图像不变，三维激光投影到二维）

ROS kitti player

1.Scan Matching

Scan-to-scan matching---ICP,etc

- 计算成本大
- 累积误差，需进行闭环
- 容易进行闭环检测

Scan-to-map matching---Hector SLAM,etc

- 误差累积小，计算成本小
- 难以闭环

2.ICP---(PL-ICP)

PL-ICP: ICP with point-to-line metric

- What happens if we use a point-to-line metric?

- Good things:
  - quadratic convergence instead of linear
  - convergence in a finite number of steps
  - much faster in practice

- Bad things:
  - less robust for large rotations

ICP variant using a point-to-line metric. ICRA2009

2.ICP---(PL-ICP)

Metrics zoo

- Vanilla ICP and PL-ICP both **use the same global function**:

$$\boxed{\min_{q} \sum_i \left\| p_i \oplus q - \Pi\{\mathcal{S}^{\text{ref}}, p_i \oplus q\} \right\|^2}$$

→ same **statistical** properties

- but a **different incremental** one:

→ different **numerical** properties

ICP: point-to-point

$$\min_{q_{k+1}} \sum_i \left\| p_i \oplus q_{k+1} - \Pi\{\mathcal{S}^{\text{ref}}, p_i \oplus q_k\} \right\|^2$$

PL-ICP: point-to-line

$$\min_{q_{k+1}} \sum_i \left(n_i^T \left[p_i \oplus q_{k+1} - \Pi\{\mathcal{S}^{\text{ref}}, p_i \oplus q_k\} \right] \right)^2$$

q	robot pose (world frame)
p_i	points in the first scan
$\Pi\{\mathcal{S}^{\text{ref}}, \cdot\}$	projection on the reference surface
$\oplus$	rototranslation: $p_i \oplus q_k = \mathbf{R}(\theta_k)\, p_i + t_k$
n_i	normal to surface

ICP variant using a point-to-line metric. ICRA2009

2.ICP---(PL-ICP)

1. 坐标变换

2. 找到两个最近点

3. 剔除异常点

4-5. 求误差函数的迭代增量

Repeat for $k \geq 0$ until convergence or loop detected:

1 – Compute the coordinates of the second scan's points in the first scan's frame of reference, according to the current guess $\boldsymbol{q}_k = (\boldsymbol{t}_k, \theta_k)$. Point $\boldsymbol{p}_i$ is transformed into $\boldsymbol{p}_i^w$ as follows:

$$\boldsymbol{p}_i^w \triangleq \boldsymbol{p}_i \oplus \boldsymbol{q}_k = \mathbf{R}(\theta_k)\,\boldsymbol{p}_i + \boldsymbol{t}_k \tag{4}$$

2 – For each point $\boldsymbol{p}_i^w$, find the two closest points in the first scan; call their indexes j_1^i e j_2^i. Call $\boldsymbol{C}_k$ all the point-to-segment correspondences at step k. $\boldsymbol{C}_k$ can be written as a set of tuples $\langle i, j_1^i, j_2^i \rangle$, meaning: point i is matched to segment j_1^i–j_2^i.

3 – Use a trimming procedure [9] to eliminate outliers.

4 – Rewrite the error function (3) as:

$$J(\boldsymbol{q}_{k+1}, \boldsymbol{C}_k) = \sum_i \left(\boldsymbol{n}_i^{\mathrm{T}} \left[\mathbf{R}(\theta_{k+1})\boldsymbol{p}_i + \boldsymbol{t}_{k+1} - \boldsymbol{p}_{j_1^i} \right] \right)^2 \tag{5}$$

This is the sum of the squares of the distances from point i to the line containing the segment j_1^i–j_2^i.

5 – To obtain $\boldsymbol{q}_{k+1}$, minimize the error function (5) using the algorithm in Appendix *I*.

ICP variant using a point-to-line metric. ICRA2009

2.ICP---(PL-ICP)

1. 坐标变换

```
//--->1投 影 到 世 界 坐 标 系
ld_compute_world_coords(laser_sens, x_old);
```

2. 找到两个最近点

```
/** Find correspondences (the naif or smart way) */
//----2寻 找 对 应 点
if(params->use_corr_tricks)
    find_correspondences_tricks(params);
else
    find correspondences(params);
```

3. 剔除异常点

```
/* Trim correspondences */
//----3剔 除 异 常 匹 配
kill_outliers_trim(params, &error);
int num_corr_after = ld_num_valid_correspondences(laser_sens);
```

4-5. 求误差函数的迭代增量

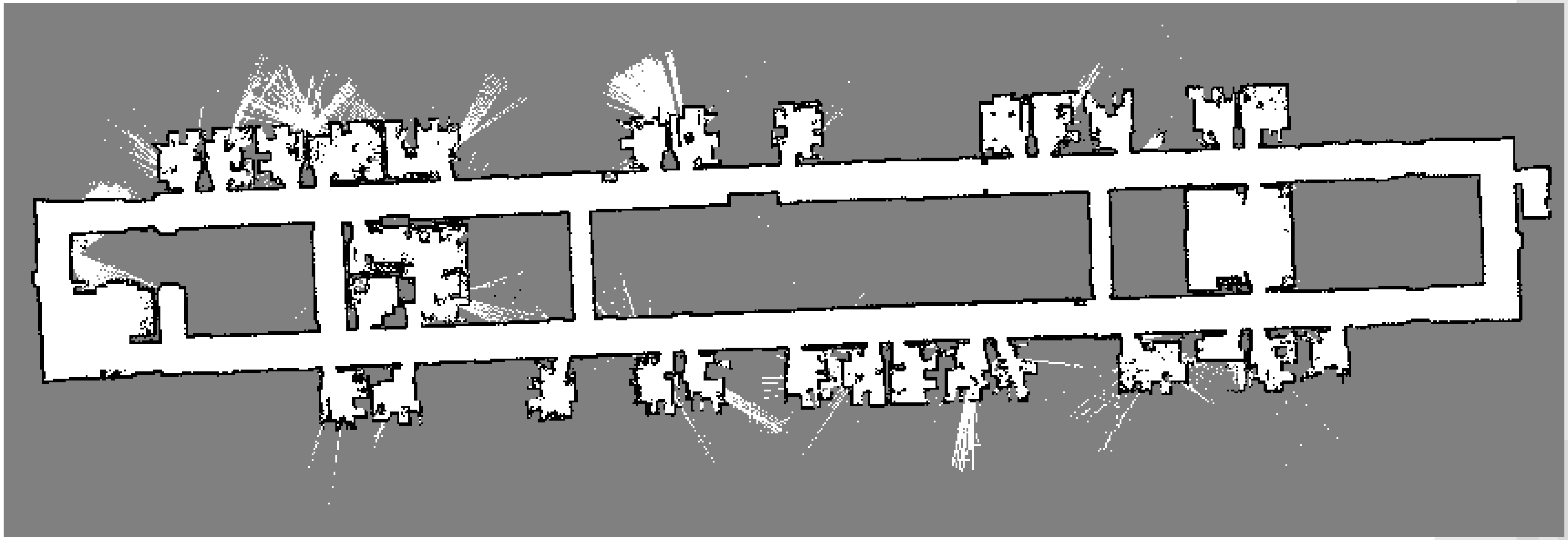
```
/* Compute next estimate based on the correspondences */
//-----4 5计 算 下 一 位 姿
if(!compute_next_estimate(params, x_old, x_new)) {
    sm_error("  icp_loop: Cannot compute next estimate.\n");
    all_is_okay = 0;
    egsl_pop_named("icp_loop iteration");
    break;
}
```

ICP variant using a point-to-line metric. ICRA2009

2.ICP---(PL-ICP)

PL-ICP加闭环(g2o)—demo
*关键帧的选择策略
*如何闭环？

仿真实验-ROS Stage
http://wiki.ros.org/stage



2.ICP---(PL-ICP)

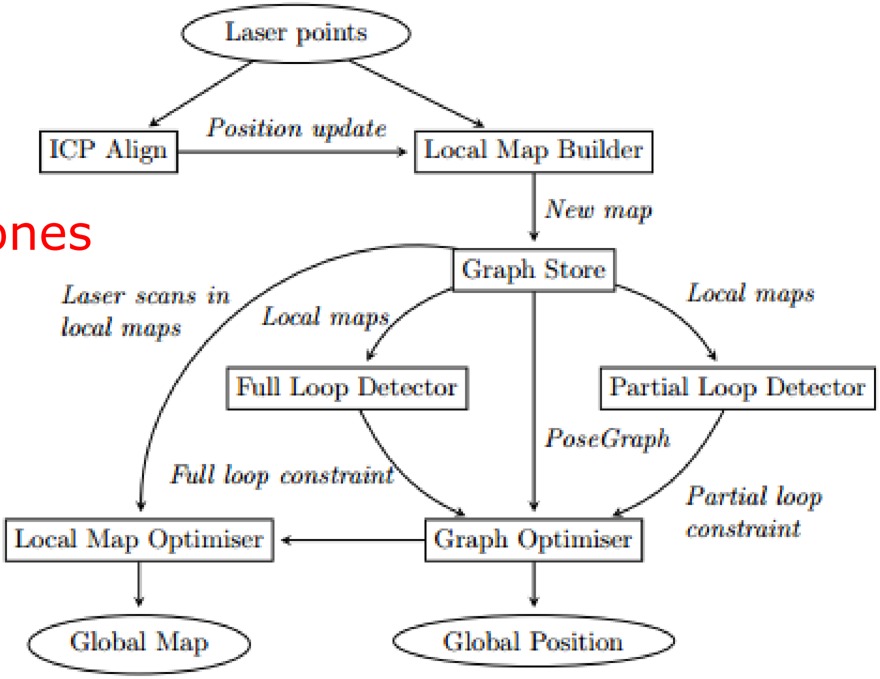
*关键帧的选择策略
*如何闭环？

Local Map Based Graph SLAM with Hierarchical Loop Closure and Optimisation.ACRA2015

无视频，无代码

Full loop closures and Partial ones

类似于ORB _SLAM



Algorithm	Final Position Error		Trajectory Error	
	Dist (m)	Angular (°)	Dist (m)	Angular (°)
Graph SLAM	0.03	6.87	0.04 ± 0.05	0.57 ± 1.15
POGMBICP	11.06	14.88	0.44 ± 1.61	0.57 ± 1.72
Graph SLAM (no partial loop closures)	0.20	6.15	0.06 ± 0.07	0.57 ± 1.15
Graph SLAM (no per scan covariance)	10.68	12.26	0.19 ± 0.42	1.72 ± 2.86
Hector SLAM	15.84	31.51	2.17 ± 3.75	14.32 ± 25.78
Graph SLAM (with Hector SLAM)	0.10	11.45	0.14 ± 0.38	4.01 ± 15.47
FastSLAM (with POGMBICP)	8.82	19.3	2.89 ± 3.09	8.02 ± 14.90
GMapping (with wheel odometry)	18.46	59.0	1.76 ± 3.04	1.72 ± 2.29
GMapping (with POGMBICP)	10.83	14.8	0.28 ± 0.59	5.16 ± 16.04

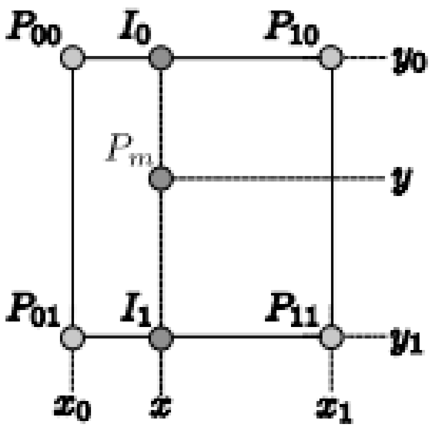
2.ICP---(PL-ICP)

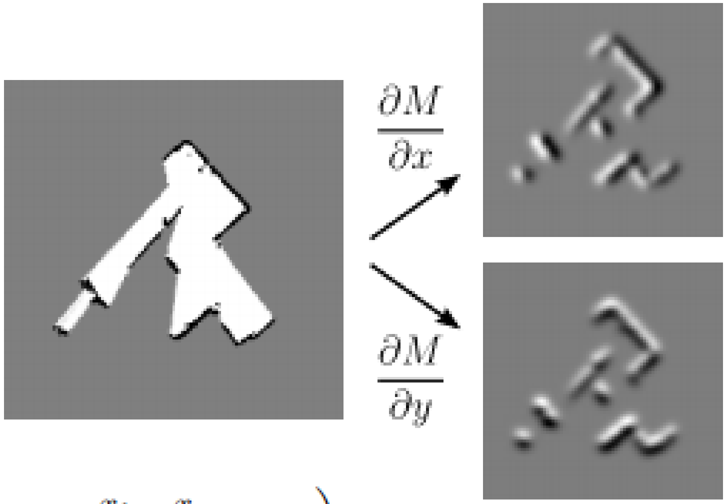
*关键帧的选择策略
*如何闭环？

Real-Time Loop Closure in 2D LIDAR SLAM. ICRA2016
来自谷歌，开源

3.Hector SLAM

地图和地图表示





$$M(P_m) \approx \frac{y-y_0}{y_1-y_0}\left(\frac{x-x_0}{x_1-x_0}M(P_{11}) + \frac{x_1-x}{x_1-x_0}M(P_{01})\right) + \frac{y_1-y}{y_1-y_0}\left(\frac{x-x_0}{x_1-x_0}M(P_{10}) + \frac{x_1-x}{x_1-x_0}M(P_{00})\right)$$

$$\frac{\partial M}{\partial x}(P_m) \approx \frac{y-y_0}{y_1-y_0}(M(P_{11}) - M(P_{01})) + \frac{y_1-y}{y_1-y_0}(M(P_{10}) - M(P_{00}))$$

$$\frac{\partial M}{\partial y}(P_m) \approx \frac{x-x_0}{x_1-x_0}(M(P_{11}) - M(P_{10})) + \frac{x_1-x}{x_1-x_0}(M(P_{01}) - M(P_{00}))$$

```
float dx1 = intensities[0] - intensities[1];
float dx2 = intensities[2] - intensities[3];

float dy1 = intensities[0] - intensities[2];
float dy2 = intensities[1] - intensities[3];

float xFacInv = (1.0f - factors[0]);
float yFacInv = (1.0f - factors[1]);

//返 回 插 值 和 微 分
return Eigen::Vector3f(
  ((intensities[0] * xFacInv + intensities[1] * factors[0]) * (yFacInv)) +
  ((intensities[2] * xFacInv + intensities[3] * factors[0]) * (factors[1])),
  -((dx1 * xFacInv) + (dx2 * factors[0])),
  -((dy1 * yFacInv) + (dy2 * factors[1]))
);
```

A Flexible and Scalable SLAM System with Full 3D Motion Estimation.SSRR2011

3.Hector SLAM

地图更新

$$\sum_{i=1}^{n}[1-M(\mathbf{S}_i(\xi+\Delta\xi))]^2 \rightarrow 0\,.$$

$$\sum_{i=1}^{n}\left[1-M(\mathbf{S}_i(\xi))-\nabla M(\mathbf{S}_i(\xi))\frac{\partial \mathbf{S}_i(\xi)}{\partial \xi}\Delta\xi)\right]^2 \rightarrow 0$$

$$2\sum_{i=1}^{n}\left[\nabla M(\mathbf{S}_i(\xi))\frac{\partial \mathbf{S}_i(\xi)}{\partial \xi}\right]^T \left[1-M(\mathbf{S}_i(\xi))-\nabla M(\mathbf{S}_i(\xi))\frac{\partial \mathbf{S}_i(\xi)}{\partial \xi}\Delta\xi)\right]=0$$

$$\Delta\xi = \mathbf{H}^{-1}\sum_{i=1}^{n}\left[\nabla M(\mathbf{S}_i(\xi))\frac{\partial \mathbf{S}_i(\xi)}{\partial \xi}\right]^T [1-M(\mathbf{S}_i(\xi))]$$

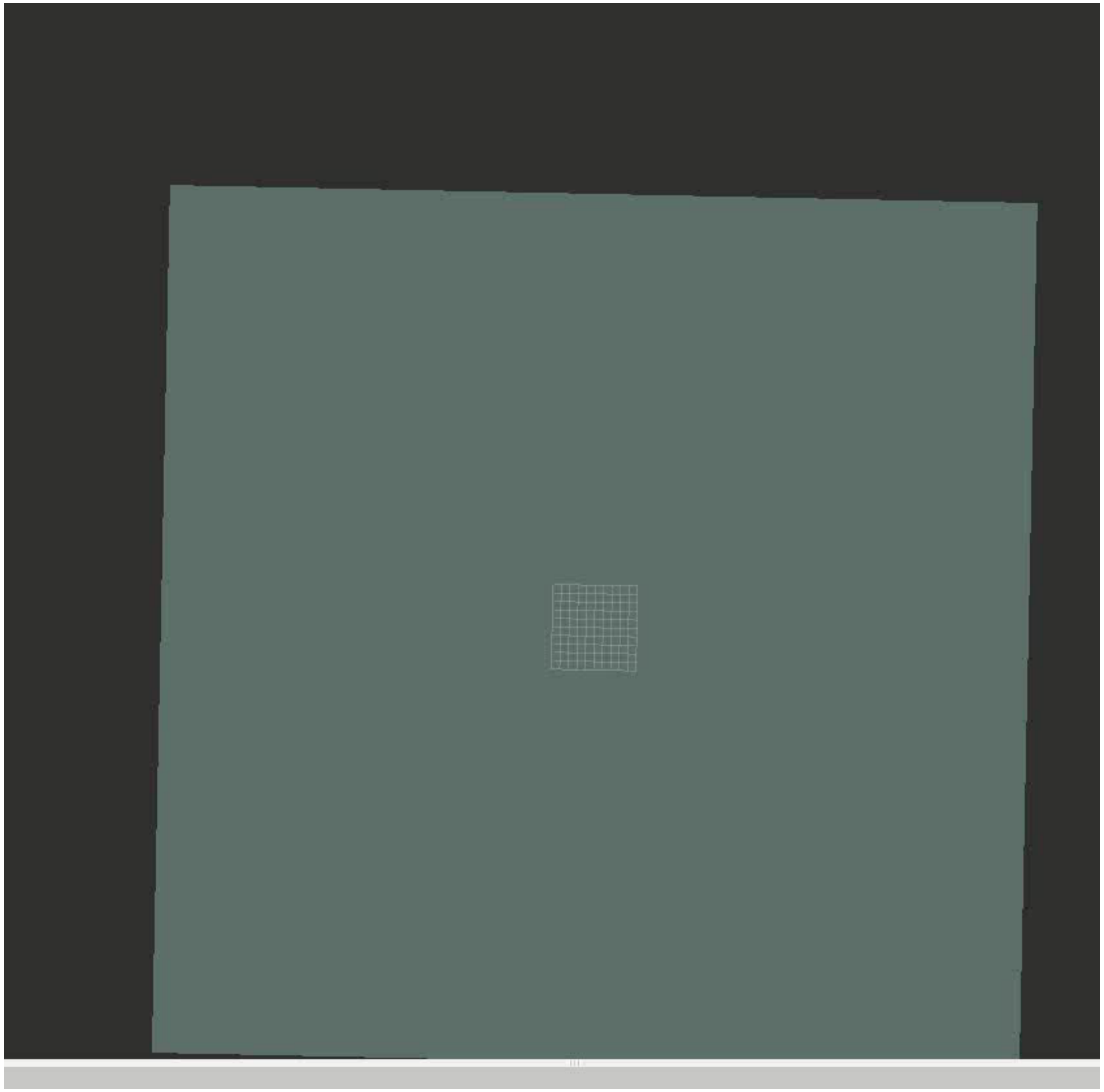
$$\mathbf{H}=\left[\nabla M(\mathbf{S}_i(\xi))\frac{\partial \mathbf{S}_i(\xi)}{\partial \xi}\right]^T \left[\nabla M(\mathbf{S}_i(\xi))\frac{\partial \mathbf{S}_i(\xi)}{\partial \xi}\right]$$

```
for (int i = 0; i < size; ++i)
{

  const Eigen::Vector2f& currPoint (dataPoints.getVecEntry(i));

  //插 值 和 微 分
  Eigen::Vector3f transformedPointData(interpMapValueWithDerivatives(transform *

  float funVal = 1.0f - transformedPointData[0];

  dTr[0] += transformedPointData[1] * funVal;
  dTr[1] += transformedPointData[2] * funVal;

  float rotDeriv = ((-sinRot * currPoint.x() - cosRot * currPoint.y()) * transfo

  dTr[2] += rotDeriv * funVal;

  H(0, 0) += util::sqr(transformedPointData[1]);
  H(1, 1) += util::sqr(transformedPointData[2]);
  H(2, 2) += util::sqr(rotDeriv);

  H(0, 1) += transformedPointData[1] * transformedPointData[2];
  H(0, 2) += transformedPointData[1] * rotDeriv;
  H(1, 2) += transformedPointData[2] * rotDeriv;
} ? end for inti=0;i<size;++i ?
H(1, 0) = H(0, 1);
H(2, 0) = H(0, 2);
H(2, 1) = H(1, 2);
```

A Flexible and Scalable SLAM System with Full 3D Motion Estimation.SSRR2011

3.Hector SLAM

Demo

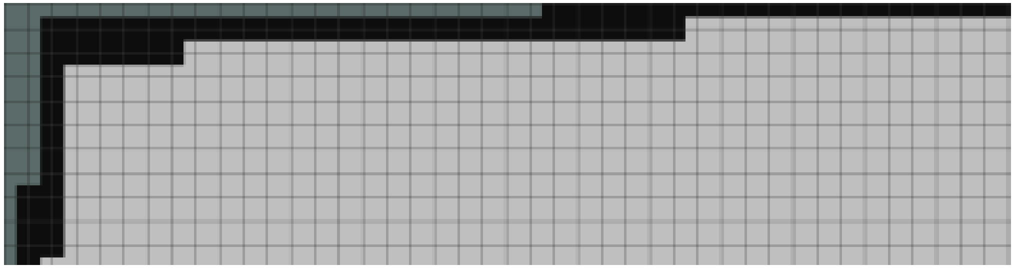


A Flexible and Scalable SLAM System with Full 3D Motion Estimation.SSRR2011

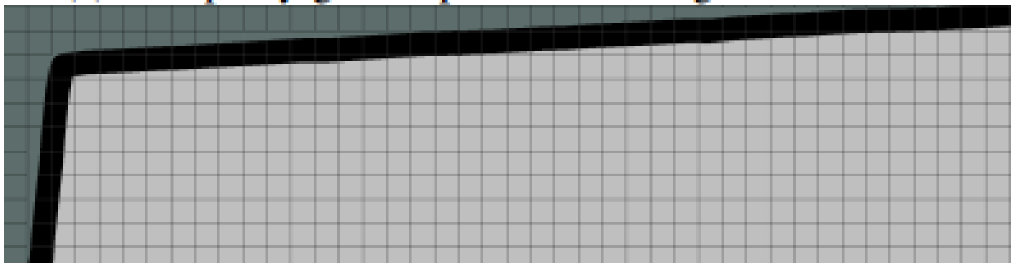
4.2D-SDF-SLAM

→Point-based map:这种方法的精度比较高，但是并没有提供环境的形状信息，并不适合于后面的路径规划与障碍物检测。
→Grid-based map:用栅格表达，但是低于栅格大小的点就没有精度了。

受Kinectfusion的启发，Sdf的方法解决了这个问题，在得到环境形状信息的基础之上，又尽量不失结果的精度。



(a) Occupancy grid maps lead to strong discretization.



(b) Same environment as in (a) represented by a signed distance function (SDF) map, as proposed by this paper.

2D-SDF-SLAM:A Signed Distance Function based SLAM Frontend for Laser Scanners.ICRA2015

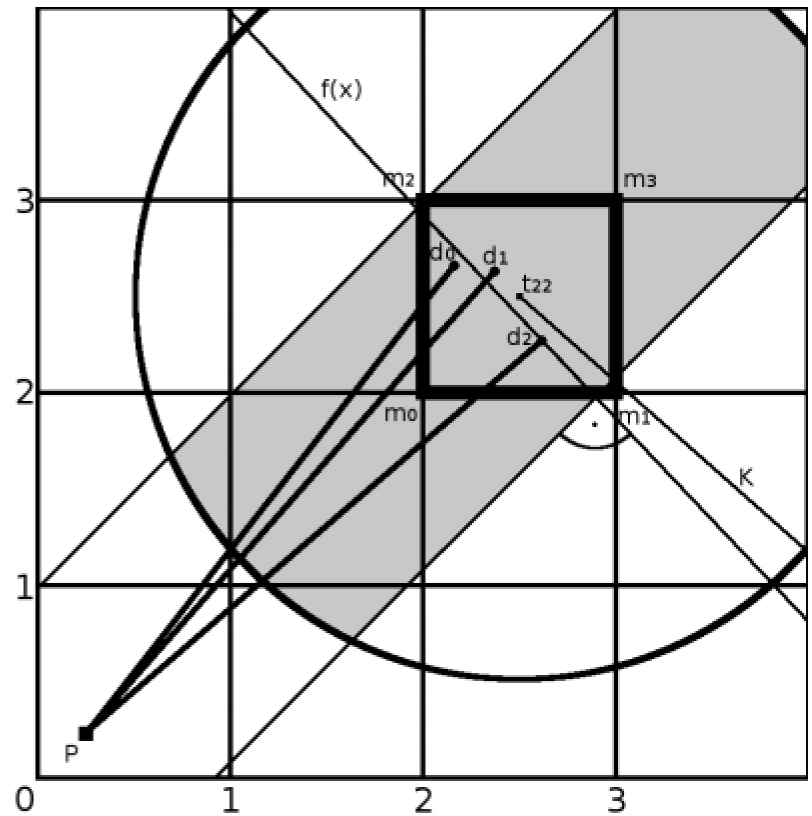
4.2D-SDF-SLAM

1. Mapping

Deming Regression

$$\frac{1}{\sigma^2}\sum_{i=1}^{I}\left((d_i^y-\beta_0-\beta_1 x_i)^2+(d_i^x-x)^2)\right)$$

```
void deming_regression(float tars[][2], int num, double &
    //todo hackhack, use neighbors
    bool revert = false;
    if (num == 0) {
        num = 2;
        revert = true;
    }

    //orthogonal deming regression
    double xbar = 0;
    double ybar = 0;
    double sxx = 0;
    double sxy = 0;
    double syy = 0;
    for (int i = 0; i < num; i++) {
        xbar += util::toMap(tars[i][0], p_grid_res_, p_map_si
        ybar += util::toMap(tars[i][1], p_grid_res_, p_map_si
    }
    xbar /= num;
    ybar /= num;
    for (int i = 0; i < num; i++) {
        sxx += (util::toMap(tars[i][0], p_grid_res_, p_map_si
        sxy += (util::toMap(tars[i][0], p_grid_res_, p_map_si
        syy += (util::toMap(tars[i][1], p_grid_res_, p_map_si
    }
    sxx /= num - 1;
    sxy /= num - 1;
    syy /= num - 1;

    beta1 = (syy - sxx + sqrt((syy - sxx) * (syy - sxx) + 4 *
    beta0 = ybar - beta1 * xbar;
    //y=beta0+beta1*x
    //end deming
```



4.2D-SDF-SLAM

2. Scan Registration

$$P^* = \arg\min_P \sum_{i=0,\dots,I} (M(P \otimes d_i'))^2.$$

$$\frac{\partial M}{\partial x}(d_i) \approx y(m_3 - m_2) + (1 - y)(m_1 - m_0),$$
$$\frac{\partial M}{\partial y}(d_i) \approx x(m_2 - m_0) + (1 - x)(m_3 - m_1),$$

$$M(d_i) \approx |y(m_3 x + m_2(1 - x)) + (1 - y)(m_1 x + m_0(1 - x))|$$

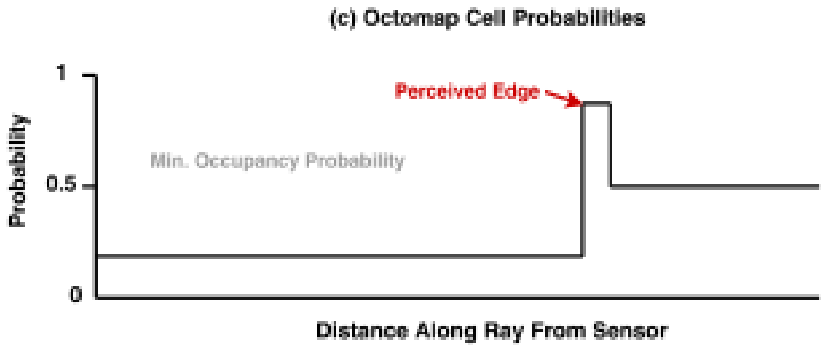
$$\sum_{i=1}^{I} [M((P' + \Delta P') \otimes d_i')]^2 \rightarrow 0.$$

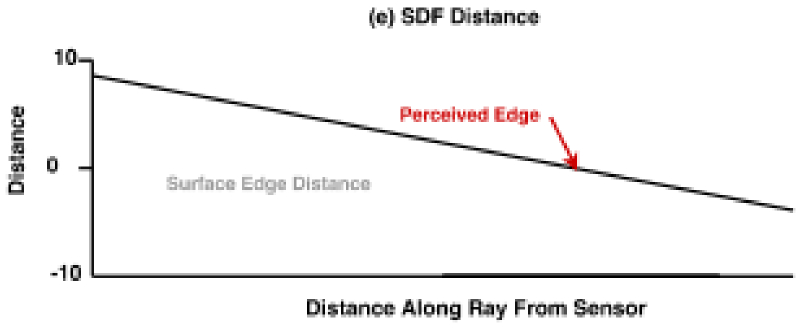
```
funVal = transformedPointData[0];
//    rotDeriv = ((-sinRot * currPoint[0] - cosRot * currPoint[1]) * transformedPointData[1] + (cos
//currPoint[0]/=0.05;
//currPoint[1]/=0.05;
rotDeriv = ((-sinRot * currPoint[0] - cosRot * currPoint[1]) * transformedPointDat
            (cosRot * currPoint[0] - sinRot * currPoint[1]) * transformedPointDat
//误 差 项
t_dTr[0] += transformedPointData[1] * funVal;
t_dTr[1] += transformedPointData[2] * funVal;
t_dTr[2] += rotDeriv * funVal;

t_H[0][0] += transformedPointData[1] * transformedPointData[1];
t_H[1][1] += transformedPointData[2] * transformedPointData[2];
t_H[2][2] += rotDeriv * rotDeriv;

t_H[0][1] += transformedPointData[1] * transformedPointData[2];
//    ROS_ERROR("test %f", t_H[0][1]);
t_H[0][2] += transformedPointData[1] * rotDeriv;
t_H[1][2] += transformedPointData[2] * rotDeriv;
end for PCLPointCloud.const ... ?
```

4.2D-SDF-SLAM

3. SDF over Grid-map

Mapping and planning often have very different requirements from an environmental representation.





Gradient-based trajectory optimization can be easily implemented

Signed Distance Fields: A Natural Representation for Both Mapping and Planning.RSS16

4.2D-SDF-SLAM

3. SDF over Grid-map

octomap三大优点
1、开源
2、概率化表示
3、时间空间高效
缺点：概率适合于雷达之类的，视觉的方法不太好，因为误差与距离有关，不是单纯的高斯模型

SDF的方法

Two values for each voxel: the distance to the surface (along the ray from the camera) and the weight/probability of this measurement.

Signed Distance Fields: A Natural Representation for Both Mapping and Planning.RSS16