KCF 算法介绍报告

姓名：　高鑫　田野

单位：寒武纪智能科技有限公司

前言

本文档分为五部分。第一部分为简介，大概叙述了 KCF 算法的思路，由于本文档着重于 KCF 算法的优化，所以简介部分相对较少，如果想要跟多的了解跟踪算法以及 KCF 可以参考（超链接 1、2、3、4、5）；第二部分为算法的理论基础，本部分涉及到的理论有回归分类，核方法，傅里叶变换，循环矩阵，卷积等等。在文件夹中放有相关的文献资料以供查阅；第三部分讨论了论文中主要的公式在代码中的对应部分，我们将不同版本的代码也放入了文件夹中以供查阅，同时附上 KCF 作者本人的主页地址（KCF 作者主页）；第四部分旨在建立参数优化的量化指标以及标准流程，目标是实现参数优化的规范化、标准化，使得优化过程有章可循。第五部分为附录，这部分简单介绍了诸如循环矩阵这些在算法理论中必须了解的概念。

本文档并非一个自我完备的文档，我们在其中添加了很多超链接，包括网址和参考文献，其中参考文献我们会连同本文档一起添加到文件夹中。除此之外，在整理 KCF 的过程中也搜集到了许多其他的参考文献。这些参考文献与本文档中的内容没有直接的关系，但我们会附到其它文件夹中，以备查阅。

我们将所有搜集到的资料附上的原因还在于，希望其他人可以对这个文档做出有价值的完善，KCF（或者说跟踪算法）本身有很多东西需要我们阐明。因此我们很期待有人能对此提出建设性的意见。

本文档由高鑫和田野合作完成，高鑫主要承担了算法基础理论理解，公式推导，参数优化流程，以及数学、信号/图像处理基础理论，文档编辑的工作。田野主要承担了 C++源码的解读工作。

目 录

第一章　简介

KCF（Kernelized Correlation Filter）即核相关滤波，它是一种基于检测的追踪算法，它的整体思路是：将图像信息通过特征提取的方式获得以后，利用循环移位等效滑动窗口(其数学模型为循环矩阵）获得丰富的训练样本（密集采样-dense sampling），而后再利用循环矩阵的一系列良好性质（主要是它在傅里叶对角化和卷积方面的良好表现）处理了来自（最优化）回归分类的结果。在这个过程中利用了核技巧(Kernel trick)将低维空间线性不可分的样本数据通过构造高维空间映射的内积（核函数）变成线性可分问题，从而完成正负样本的分类。它的创新之处就在于：利用循环移位巧妙地构造出了丰富的样本，然后在整个计算过程中反复使用傅里叶变换，将复杂的矩阵运算简化为简单的智能点积运算，从而使运算复杂度大大降低。它的难点在于：由于核技巧所应用到的数学理论极其抽象，对于这一部分的理解是整个算法中最大的难点。

表 1. KCF 算法难点

难点	具体描述	性质	特点	可能的解决方案
核技巧	通过核函数技巧将线性不可分问题变为线性可分	数学理论	理论难点；相关文献比较丰富；理论体系本身比较成熟	通过学习相关理论可以解决（模式分析的核方法_en.pdf P25）

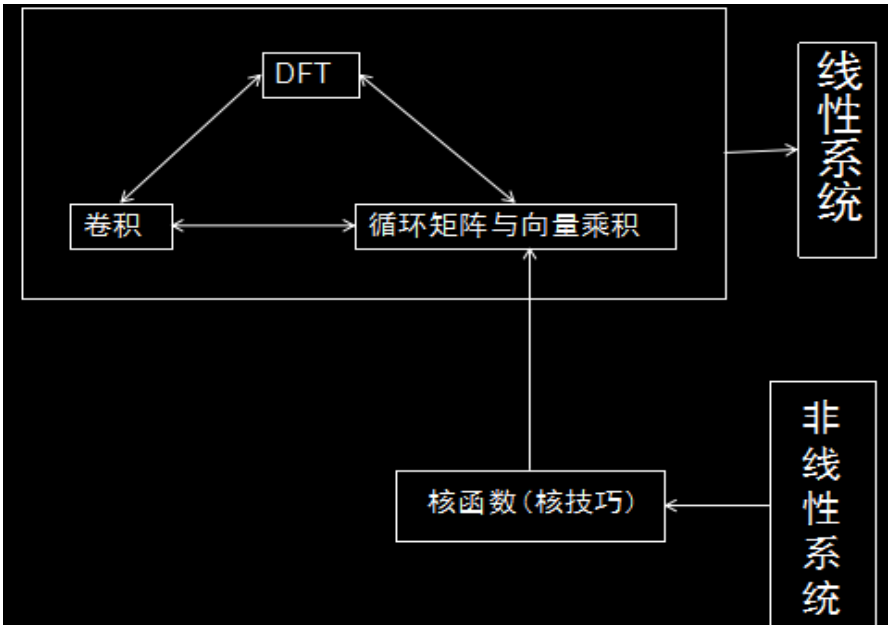


图 1. KCF 算法思路

第二章 算法理论

2.1 （最优化）回归

追踪就是根据初始框选区域或前一帧图片中的目标适当放大以后的区域（padding）为样本，然后利用样本按照一定的准则（KCF 中为最小正则化风险）进行最优化回归，得到回归系数（回归器的各种参数，它们确定了回归器）。可以想象，既然回归系数是在以目标为样本的基础上得到的，那么在对下一帧图片的检测中我们有理由认为在目标所在区域回归器会出现最大响应，于是我们将下一帧图片中响应最大的区域判定为目标所在区域，最后完成追踪。

2.1.1 线性部分

考虑如下最小正则化风险的优化问题：

$$\min_{\mathbf{w}} \sum_i (f(\mathbf{x}_i)-y_i)^2+\lambda\|\mathbf{w}\|^2 \qquad (1)$$

$\mathbf{x}_i$ —样本区域；

y_i —采集到的样本值；

λ —正则化系数；

f —要训练的回归器，$f(\mathbf{x})=\mathbf{w}^T\mathbf{x}$ (1.1)

令$l=(X\mathbf{w}-\mathbf{y})^T(X\mathbf{w}-\mathbf{y})+\lambda\mathbf{w}^T\mathbf{w}$ 其中，$X=(\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_n)^T,\mathbf{y}=(y_1,y_2,\cdots,y_n)^T$

则上面的问题有如下形式的解：

$$\frac{\partial l}{\partial \mathbf{w}}=2X^TX\mathbf{w}-2X^T\mathbf{y}+2\lambda\mathbf{w}=\mathbf{0}$$

即就是 ,$(X^TX+\lambda I)\mathbf{w}=X^T\mathbf{y}\Leftrightarrow$

$$\mathbf{w}=(X^TX+\lambda I)^{-1}X^T\mathbf{y} \qquad \cdots\cdots(2)$$

==========注=====解==========

1、关于为何要最小化 w，请参见如下两篇博文：

http://my.oschina.net/wangguolongnk/blog/111349#navbar-header

（3、最大间隔化）；

http://blog.csdn.net/macyang/article/details/38782399/

2、如果还要了解更多与与此相关的内容可以参考【参考文献】；

3、λ的引入是为了防止过拟合（overfit,参见百度词条“过拟合”），也就是说这样做的目的是出于数值稳定性的考虑“【参考文献】”（P20，2.2.2 节）；

4、向量值函数求导请参见“《矩阵分析》 刘丁酉著 武汉大学出版社”（P200,6.4 节 6.4.1 部分正文，以及例 1、例 2）；

5、关于内积与范数的定义以及二者之间的关系请参考“《矩阵分析》 刘丁酉著 武汉大学出版社”（P119-P120，定义 4.1.1、4.1.2、4.1.3；P177-P178，定义 6.1.1，例 1）

==========

2.1.2 对偶

对于（1）式我们可以证明 $\mathbf{w}$ 可以表示为 $\mathbf{x}_i$ 的线性组合，即：

$$\mathbf{w}=\sum_{i}^{n}\alpha_i\mathbf{x}_i=X^T\boldsymbol{\alpha} \qquad (3)$$

（证明参考“henriques phd”P21）！！！注意，此式中 X^T 以行存储数据！！！

因为 X 是样本数据，我们已经知道，所以求 $\mathbf{w}$ 的过程就转化为求 $\boldsymbol{\alpha}$ 的过程。这样的转换在凸优化中称为由原问题向对偶问题的转换（详情参见“模式分析的核方法 .pdf”P31，2.2.2 Ridge Regression: Primal and Dual）。因为 KCF 算法中的原问题与对偶问题是等价问题，因此求解 $\boldsymbol{\alpha}$ 与原问题是等价的。$\boldsymbol{\alpha}$ 的表达式为：

$$\boldsymbol{\alpha}=(XX^T+\lambda I)^{-1}\mathbf{y} \qquad (4)$$

（“henriques_phd”P22）

式中 XX^T 称为格拉姆矩阵。

==========注==========解==========

关于凸优化的拓展阅读，可以参考凸优化_Convex_Optimization.pdf

==========

2.1.3 非线性

在 2.1.1 节中，我们讨论了在线性情况下的分类问题，但是往往我们所面临的分类是线性不可分的问题，这个时候就需要将分类对象映射到高维空间之中，使之在高维空间中是线性可分的（见 图 2.1.3）。也就是说，我们要定义如下

映射：

$$\mathbf{x} \to \varphi(\mathbf{x})$$

这里，$\varphi(\mathbf{x})$ 是比 $\mathbf{x}$ 维度更高的数据，于是，结合式（3），式（1.1）可以被写成如下形式：

$$\hat{y} = f(\mathbf{z}) = \sum_i \alpha_i \varphi^T(\mathbf{x}_i)\varphi(\mathbf{z}) \qquad (5)$$

大多数情况下要找到这个映射的显示表达式是困难的，也是不必要的。原因在于：我们的最终目的是为了分类，将数据点映射到高维空间只是手段不是目的，也就是说，我们最后的分类不是直接使用映射后的高维空间数据点就能完成。事实上，我们利用的是映射到高维空间的数据点之间的内积（$\varphi^T(\mathbf{x}_i)\varphi(\mathbf{z})$，见（5）式）来进行分类的。

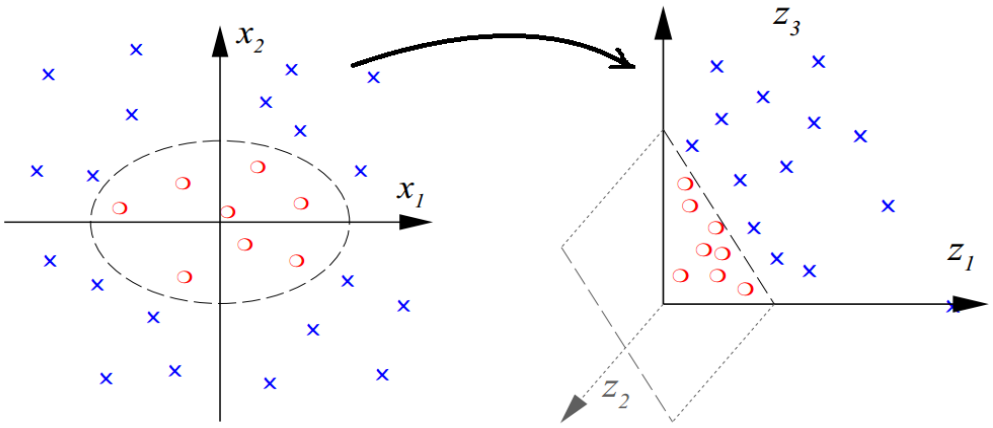


图 2.1.3　左边为二维情况下线性不可分的点（无法用线性方程对两类点划分边界），右边是将左边的点映射到高维空间后将其变成线性可分的两类点。

假如我们可以找到一个函数：

$$k(\mathbf{x}_i, \mathbf{x}_j) : \mathbf{x}_i, \mathbf{x}_j \mapsto \varphi^T(\mathbf{x}_i)\varphi(\mathbf{x}_j)$$

那么，我们就没有必要直接寻找非线性映射 $\varphi(\bullet)$ 的显示格式。

一个自然而然的问题就是什么样的函数可以满足要求，使得 $k(\mathbf{x}_i, \mathbf{x}_j) = \varphi^T(\mathbf{x}_i)\varphi(\mathbf{x}_j)$ 成立？关于这个问题的最一般解答需要参考（模式分析的核方法_en.pdf），一般情况下，满足 $\forall \mathbf{x}_i, \mathbf{x}_j$，都有 $k(\mathbf{x}_i, \mathbf{x}_j) \geq 0$ 的函数 $k(\mathbf{x}_i, \mathbf{x}_j)$ 均可以作为核函数（Mercer 核）。在 KCF 的论文中使用了最常见的高斯核

$$K(\mathbf{x},\mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2/2\sigma^2}$$

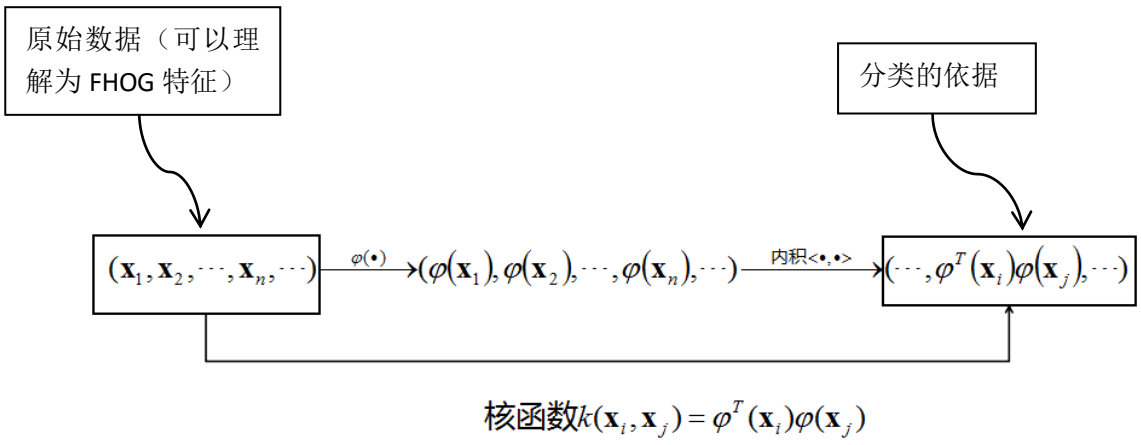


图 2.1.4　在实际操作中，我们只需要找到核函数 k 使得 $k(\mathbf{x}_i,\mathbf{x}_j)=\varphi^T(\mathbf{x}_i)\varphi(\mathbf{x}_j)$ 即可完成对分类依据（高维空间的内积）的计算

我们使用核函数以原始数据（低维空间，线性不可分）为输入，以高维空间中线性可分数据的内积（分类依据）为输出，实现了从原始数据直接到分类依据的飞跃。

有了核函数的概念，我们就可以将（5）式写为如下形式：

$$\hat{y} = f(\mathbf{z}) = \sum_i \alpha_i k(\mathbf{x}_i,\mathbf{z}) = \mathbf{k}^{\mathbf{xz}}\boldsymbol{\alpha}$$　（注意：$\mathbf{k}^{\mathbf{xz}}$ 在这里为行向量）

2.2 循环特性的加盟

==========注=======解==========

1、关于循环矩阵的定义和属性请参考附录 A 中的内容；

2、以下几篇文献中都有提及循环矩阵的相关信息以供查询：

henriques_phd.pdf P31；

Exploiting the Circulant Structure of.pdf P11；

R. M. Gray, Toeplitz and Circulant Matrices A Review. Now.pdf P185。

（这篇文献初始页码不是 P1，P185 仅仅是文档中标注的页码！！）

==========

在 KCF 中，一个创举性的工作就是通过循环移位得到一个密集的采样（dense sample）样本，这样一来既避免了随机采样带来的样本缺失，又避免了传统的滑

动窗口密集采样耗费的大量时间。

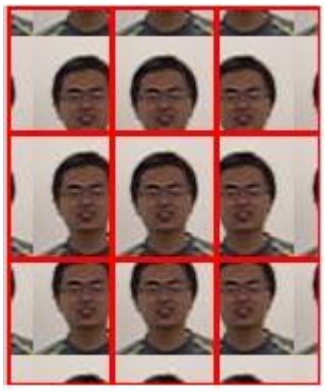


图 2.2.1　通过循环移位进行密集采样

正如图 2.2.1 所示，我们可以通过一幅图像对其进行循环移位得到丰富的样本，这样就避免了采用滑动窗口进行密集采样的弊端。需要注意的是，循环移位得到的图像在边缘是不连续的，这一点在论文中作者是通过余弦窗（关于余弦窗请参考百度词条或者冈萨雷斯的《数字图像处理》）做预处理来改善的。

回忆式（4）$\boldsymbol{\alpha}=(XX^T+\lambda I)^{-1}\mathbf{y}$，注意到这里 X 是以每一行的形式存储样本 $\mathbf{x}_i$ 的，即就是：

$$\begin{bmatrix} <\mathbf{x}_0,\mathbf{x}_0> & <\mathbf{x}_0,\mathbf{x}_1> & <\mathbf{x}_0,\mathbf{x}_2> & \cdots & <\mathbf{x}_0,\mathbf{x}_{n-1}> \\ <\mathbf{x}_1,\mathbf{x}_0> & <\mathbf{x}_1,\mathbf{x}_1> & <\mathbf{x}_1,\mathbf{x}_2> & \cdots & <\mathbf{x}_1,\mathbf{x}_{n-1}> \\ <\mathbf{x}_2,\mathbf{x}_0> & <\mathbf{x}_2,\mathbf{x}_1> & <\mathbf{x}_2,\mathbf{x}_2> & \cdots & <\mathbf{x}_2,\mathbf{x}_{n-1}> \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ <\mathbf{x}_{n-1},\mathbf{x}_0> & <\mathbf{x}_{n-1},\mathbf{x}_1> & <\mathbf{x}_{n-1},\mathbf{x}_2> & \cdots & <\mathbf{x}_{n-1},\mathbf{x}_{n-1}> \end{bmatrix}$$

上式中，$<\bullet,\bullet>$ 为内积，即 $<\mathbf{x}_i,\mathbf{x}_j>=\mathbf{x}_i^T\mathbf{x}_j$

结合上一节非线性部分（2.1.3），将 $\mathbf{x}_i$ 映射到 $\varphi(\mathbf{x}_i)$ 于是 XX^T 对应的高维映射为：

$$\begin{bmatrix} <\varphi(\mathbf{x}_0),\varphi(\mathbf{x}_0)> & <\varphi(\mathbf{x}_0),\varphi(\mathbf{x}_1)> & <\varphi(\mathbf{x}_0),\varphi(\mathbf{x}_2)> & \cdots & <\varphi(\mathbf{x}_0),\varphi(\mathbf{x}_{n-1})> \\ <\varphi(\mathbf{x}_1),\varphi(\mathbf{x}_0)> & <\varphi(\mathbf{x}_1),\varphi(\mathbf{x}_1)> & <\varphi(\mathbf{x}_1),\varphi(\mathbf{x}_2)> & \cdots & <\varphi(\mathbf{x}_1),\varphi(\mathbf{x}_{n-1})> \\ <\varphi(\mathbf{x}_2),\varphi(\mathbf{x}_0)> & <\varphi(\mathbf{x}_2),\varphi(\mathbf{x}_1)> & <\varphi(\mathbf{x}_2),\varphi(\mathbf{x}_2)> & \cdots & <\varphi(\mathbf{x}_2),\varphi(\mathbf{x}_{n-1})> \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ <\varphi(\mathbf{x}_{n-1}),\varphi(\mathbf{x}_0)> & <\varphi(\mathbf{x}_{n-1}),\varphi(\mathbf{x}_1)> & <\varphi(\mathbf{x}_{n-1}),\varphi(\mathbf{x}_2)> & \cdots & <\varphi(\mathbf{x}_{n-1}),\varphi(\mathbf{x}_{n-1})> \end{bmatrix}$$

由核函数的定义（见 2.1.3 节），上式可以写为：

$$K=\begin{bmatrix} k(\mathbf{x}_0,\mathbf{x}_0) & k(\mathbf{x}_0,\mathbf{x}_1) & k(\mathbf{x}_0,\mathbf{x}_2) & \cdots & k(\mathbf{x}_0,\mathbf{x}_{n-1}) \\ k(\mathbf{x}_1,\mathbf{x}_0) & k(\mathbf{x}_1,\mathbf{x}_1) & k(\mathbf{x}_1,\mathbf{x}_1) & \cdots & k(\mathbf{x}_1,\mathbf{x}_{n-1}) \\ k(\mathbf{x}_2,\mathbf{x}_0) & k(\mathbf{x}_2,\mathbf{x}_1) & k(\mathbf{x}_2,\mathbf{x}_2) & \cdots & k(\mathbf{x}_2,\mathbf{x}_{n-1}) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_{n-1},\mathbf{x}_0) & k(\mathbf{x}_{n-1},\mathbf{x}_1) & k(\mathbf{x}_{n-1},\mathbf{x}_2) & \cdots & k(\mathbf{x}_{n-1},\mathbf{x}_{n-1}) \end{bmatrix} \quad (6)$$

我们在本节一开始已经提到，样本是通过循环移位得到的，如附录 A1.1.2 所提到的那样，我们可以有 $\mathbf{x}_0$ 循环移位得到 $\mathbf{x}_i = P^i\mathbf{x}_0$。这样，对于矩阵 K 中的元素 $K_{ij} = k(\mathbf{x}_i,\mathbf{x}_j)$ 可以写为：

$$K_{ij} = k(\mathbf{x}_i,\mathbf{x}_j) = k(P^i\mathbf{x}_0, P^j\mathbf{x}_0)$$

在 KCF 中，我们选取的核函数是高斯核 $K(\mathbf{x},\mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2/2\sigma^2}$，这样的核函数有一个重要的性质：

对于循环矩阵 $C(\mathbf{x})$ 中的元素 $\mathbf{x}_i,\mathbf{x}_j$ 它们的核函数值满足以下条件：

$$k(M\mathbf{x}_i, M\mathbf{x}_j) = k(\mathbf{x}_i,\mathbf{x}_j) = k(P^i\mathbf{x}_0, P^j\mathbf{x}_0) \quad (7)$$

上式中 M 为置换矩阵，即 $M = P^k, k \in z$

有（7）可知：

$$K_{i,j} = k(\mathbf{x}_i,\mathbf{x}_j) = k(P^i\mathbf{x}_0, P^j\mathbf{x}_0) = k(\mathbf{x}_0, P^{j-i}\mathbf{x}_0) = k(\mathbf{x}_0, P^{(j-i)\bmod n}\mathbf{x}_0)$$

于是，由循环矩阵的定义可知：

矩阵 K 为循环矩阵。于是我们将 K 的第一行记为 $\mathbf{k}^{xx}$，那么 K 就可以记为 $C(\mathbf{k}^{xx})$。

2.3 离散傅里叶变换（DFT）

==========注=====解==========

关于 DFT 本身定义及性质，需要参考附录 B；

关于 DFT 与循环矩阵的特性需要参考附录 A2.2

==========

对于（4）式 $\boldsymbol{\alpha} = (K+\lambda I)^{-1}\mathbf{y}$，因为矩阵 K 是循环矩阵，由循环矩阵的性质（A.2.1）可以知道 $(K+\lambda I)^{-1}$ 也是循环矩阵，所以 $\boldsymbol{\alpha}$ 可以用傅里叶变换计算：

$$\hat{\boldsymbol{\alpha}} = \wp(\boldsymbol{\alpha}) = \wp\{C[\wp^{-1}(\frac{1}{\hat{\mathbf{k}}^{\mathbf{xx}} + \lambda})]\mathbf{y}\} = \frac{\hat{\mathbf{y}}}{\hat{\mathbf{k}}^{\mathbf{xx}} + \lambda}$$

到这里，我们已经完成了训练阶段的工作。

2.4 检测

在检测部分，我们针对测试图像 $\mathbf{z}$ 的所有（循环移位）样本 $\mathbf{z}_i = P^i\mathbf{z}\ (\mathbf{z}_0 = \mathbf{z})$ 进行分类，于是有：

$$f(\mathbf{z}_i) = \sum_j \alpha_j k(P^i\mathbf{z}, P^j\mathbf{x})$$

令，

$$\mathbf{f}(\mathbf{z}) = [f(\mathbf{z}_0), f(\mathbf{z}_1), f(\mathbf{z}_2), \cdots, f(\mathbf{z}_{n-1})]^T$$

那么，有：

$$\mathbf{f}(\mathbf{z}) = \begin{bmatrix} f(\mathbf{z}_0) \\ f(\mathbf{z}_1) \\ \vdots \\ f(\mathbf{z}_{n-1}) \end{bmatrix} = \begin{bmatrix} f(\mathbf{z}) \\ f(P\mathbf{z}) \\ \vdots \\ f(P^{n-1}\mathbf{z}) \end{bmatrix} = \begin{bmatrix} \sum_j \alpha_j k(\mathbf{z}, \mathbf{x}_j) \\ \sum_j \alpha_j k(P\mathbf{z}, \mathbf{x}_j) \\ \vdots \\ \sum_j \alpha_j k(P^{n-1}\mathbf{z}, \mathbf{x}_j) \end{bmatrix} \overset{k(\mathbf{x}', \mathbf{x}_j) = k(\mathbf{x}', P^j\mathbf{x})}{=\!=\!=} \begin{bmatrix} \sum_j \alpha_j k(\mathbf{z}, P^j\mathbf{x}) \\ \sum_j \alpha_j k(P\mathbf{z}, P^j\mathbf{x}) \\ \vdots \\ \sum_j \alpha_j k(P^{n-1}\mathbf{z}, P^j\mathbf{x}) \end{bmatrix}$$

$$\overset{k(\mathbf{x}, \mathbf{x}') = k(U\mathbf{x}, U\mathbf{x}')}{=\!=\!=} \begin{bmatrix} \sum_j \alpha_j k(\mathbf{z}, P^j\mathbf{x}) \\ \sum_j \alpha_j k(\mathbf{z}, P^{j-1}\mathbf{x}) \\ \vdots \\ \sum_j \alpha_j k(\mathbf{z}, P^{j-n+1}\mathbf{x}) \end{bmatrix} = \begin{bmatrix} k(\mathbf{z}, \mathbf{x}) & k(\mathbf{z}, P\mathbf{x}) & k(\mathbf{z}, P^2\mathbf{x}) & \cdots & k(\mathbf{z}, P^{n-1}\mathbf{x}) \\ k(\mathbf{z}, P^{n-1}\mathbf{x}) & k(\mathbf{z}, \mathbf{x}) & k(\mathbf{z}, P\mathbf{x}) & \cdots & k(\mathbf{z}, P^{n-2}\mathbf{x}) \\ k(\mathbf{z}, P^{n-2}\mathbf{x}) & k(\mathbf{z}, P^{n-1}\mathbf{x}) & k(\mathbf{z}, \mathbf{x}) & \cdots & k(\mathbf{z}, P^{n-3}\mathbf{x}) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{z}, P\mathbf{x}) & k(\mathbf{z}, P^2\mathbf{x}) & k(\mathbf{z}, P^3\mathbf{x}) & \cdots & k(\mathbf{z}, \mathbf{x}) \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{n-1} \end{bmatrix}$$

$$= \begin{bmatrix} k_0^{\mathbf{xz}} & k_1^{\mathbf{xz}} & k_2^{\mathbf{xz}} & \cdots & k_{n-1}^{\mathbf{xz}} \\ k_{n-1}^{\mathbf{xz}} & k_0^{\mathbf{xz}} & k_1^{\mathbf{xz}} & \cdots & k_{n-2}^{\mathbf{xz}} \\ k_{n-2}^{\mathbf{xz}} & k_{n-1}^{\mathbf{xz}} & k_0^{\mathbf{xz}} & \cdots & k_{n-3}^{\mathbf{xz}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ k_1^{\mathbf{xz}} & k_2^{\mathbf{xz}} & k_3^{\mathbf{xz}} & \cdots & k_0^{\mathbf{xz}} \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{n-1} \end{bmatrix} = K^{\mathbf{xz}}\boldsymbol{\alpha} = C(\mathbf{k}^{\mathbf{xz}})\boldsymbol{\alpha}$$

注：关于上式的推导，我们与论文中有出入。在原文中核矩阵有转置（原文（21）式），而根据作者自己的意思我们推导得到的结果没有转置，在代码中也没有使用转置。通过与作者的交流我们最后认定此处应该没有转置。

得到 $\mathbf{f}(\mathbf{z})$ 以后，我们需要找到相应最大的分量以及它的位置即可判断目标在下一帧图像中的位置。

第三章 代码

3.1 图像读取

Step1.框选目标区（以下我们将目标区统称为目标框）。

Step2.加 padding（以下我们将加 padding 后的区域称为搜索区）

==========注======解==========

所谓 padding 如下图所示，最内部的实线框即为目标区，虚线框为 padding（搜索区），外部实线框为图像边界

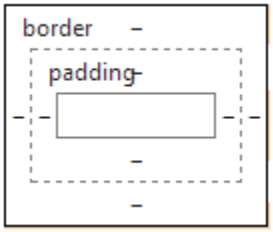


图 3.1 padding 示意图

padding 的加入是基于目标运动不是很快假设的，这样做带来了如下的后果：

A.好的一方面，将问题简化，我们可以认为目标始终存在于 padding 内。如此一来，下一帧图像中的目标无非是当前这帧图像中目标所有循环移位中的一种。

B.不好的一面，对于剧烈运动的物体 KCF 显的无能为力。

==========

3.2 特征提取（HOG）

HOG(梯度方向直方图，Histogram of Oriented Gradient）特征是描述图像的一种常用特征（关于 HOG 的介绍可以百度），它源自 2005 年的一篇文献，时至今日已经有很多 HOG 的变种，在这里我们介绍 KCF 作者使用的 FHOG 特征（由 Felzenszwalb 提出，详情参考 DPM、博客。

==========注=====解==========

（HOG 特征源码下载：

http://www.codeforge.cn/s/0/HOG%E7%89%B9%E5%BE%81-C）

==========

3.2.1 FHOG 的处理

当 FHOG 提取完毕以后将是一个 $w_b \times h_b \times 31$ 的阵列。注意！这里的 w_b、h_b 分别是搜索区宽高除以 FHOG 的 cell size（KCF 中为 4）得到的。

前面已经提到算法中通过循环移位得到丰富的样本，但这样做会使图像边缘不连续，为了减小由此带来的不良影响，代码中对 FHOG 做了预处理。即就是用余弦窗（余弦窗是这样的窗函数，它突出中心，减弱边缘，使得边缘的权重为零，以此消除不连续）对 FHOG 做了一次滤波。

3.3 分类标签

在 KCF 中，分类标签（代码中的 gaussian_shaped_labels, 论文中的 y 向量）与支持向量机中传统的二值分类不同，而是一个高斯分布函数，即中间大，四周小的轮廓。这也暗示着 KCF 假设下一帧图片中目标依然在附近（即就是目标运动不会很剧烈）。

==========注===解==========

函数 gaussian_shaped_labels 的代码如下：

```
cv::Mat KCF_Tracker::gaussian_shaped_labels(double sigma, int dim1, int dim2)
    {
        cv::Mat labels(dim2, dim1, CV_32FC1);
        int range_y[2] = {-dim2 / 2, dim2 - dim2 / 2};
        int range_x[2] = {-dim1 / 2, dim1 - dim1 / 2};
        double sigma_s = sigma*sigma;
        for (int y = range_y[0], j = 0; y < range_y[1]; ++y, ++j){
            float * row_ptr = labels.ptr<float>(j);
            double y_s = y*y;
```

```
            for (int x = range_x[0], i = 0; x < range_x[1]; ++x, ++i){
                row_ptr[i] = std::exp(-0.5 * (y_s + x*x) / sigma_s);
            }
        }
        //rotate so that 1 is at top-left corner (see KCF paper for explanation)
        cv::Mat rot_labels = circshift(labels, range_x[0], range_y[0]);
        //sanity check, 1 at top left corner
        assert(rot_labels.at<float>(0,0) >= 1.f - 1e-10f);
        return rot_labels;
```

上面的函数给出了一个以 padding 中心为原点的高斯分布。并且为后面的傅里叶变换做了预处理（调用 circshift 函数，对四个象限做对调，类似于 MATLAB 中的 fftshift 操作）。首先得到 padding 坐标以后先将其中心设为原点（这里的坐标是以 FHOG 的 cell 大小为单位的，也就是每四个像素为一个坐标单位），然后对其遍历赋值，形成高斯分布。

==========

3.4 高斯核相关

正如我们在 2.1.3 节中提到的一样，我们需要利用核函数来表征高维数据。自然，数据间的相关性也可以，并且需要用核函数表示。

在代码中（函数名为 gaussian_correlation，该函数参数名过多不在这里展开），是先对 FHOG 做了傅里叶变换，然后再求的高斯核相关。因为傅里叶变换是保范数的变换（帕萨瓦尔定理，Parseval's Theorem），所以直接利用傅里叶变换求高斯核是可行的（henriques_phd.pdf P52 式 4.15）

值得注意的是，在 C++中一个常数与向量或矩阵是可以直接相加的，其结果是给向量或矩阵的每一个元素都加上这个常数。所以对于（henriques_phd.pdf P52 式 4.15）中常数与向量（代码中是矩阵）相加减的情况应该如此理解。

在求得自相关和互相关以后，我们就可以表征分类器（代码中为变量 alphaf，论文中为$\boldsymbol{\alpha}$），然后计算响应（代码中为变量 response，论文中为$\mathbf{f}(\mathbf{z})$），最后确定最大响应及其位置（代码中调用了函数 minmaxloc()），然后对分类器和目标位置作更新（代码中采用线性插值的方法完成），完成一次循环。

第四章 参数优化

关于参数优化：

优化参数就是在不同的参数组合情况下寻找一种或几种表现相对较好的组合情况。它牵扯到两个方面：

第一，参数组合的选取；

第二，度量算法表现的可量化指标。

对于参数组合的选取，我们需要考虑的是什么样的参数对算法的性能表现有至关重要的影响？为此我们有必要对各参数的意义和作用作出分析；

对于算法表现的量化度量，首先我们要考虑两个矛盾的指标，即就是“又准又快”，通常情况下这两个指标（快和准）很难一起优化。为此我们的工作是对于不同的指标在不同应用场景下对其依照优先程度排序，也就是什么时候注重速度，什么时候精度又很重要？在优化时，首先满足优先指标然后兼顾次要指标以此类推。其次，对于已经排好序的优化指标，我们如何将其量化？这一步工作需要我们仔细挑选合适的数学量，使其尽量全面而又准确的刻画衡量我们所感兴趣的方面。

本章主要内容如下：

1、阐述代码中的各主要参数的意义和作用；

2、结合机器人视觉跟踪的实际应用背景，提出度量算法性能的指标，并对各指标按优先度排序，然后确定其量化数学描述，以此建立算法的优化准则。

4.1 主要的参变量及其意义

表 4.1 KCF 算法/代码中的各主要参量及其意义

参量	Padding（搜索区大小）	Interploation-factor (插值因子）	λ（过拟合因子）
意义	该参数提供了目标区的搜索范围	算法中分类器的更新速度	防止分类器过度学习（过拟合）
作用	参数越大，算法对快速运动的抵抗能力更强健	参数越大更新越快，对快速运动越有利，但会牺牲稳定性	太小会出现过拟合，太大会降低训练样本的意义

4.2 算法性能指标及其优先度

4.2.1 各性能指标

一般的优化都是基于数据集来做的，上面已经提到我们优化的两大指标分为“快”和“准”，也就是我们所说的速度和精度。

对于速度，最直接的方式就是帧率，这个量可以直接获得，无需多言。

对于精度，我们可以将算法的结果与标准结果（一般由人工手动操作形成）进行对比。对跟踪算法而言，我们对比的指标应该至少含有以下两项：

第一，每一帧图片中算法给出的框选区域（简称“测试框”，下同）位置与标准结果（简称“标准框”，下同）在图片中的位置差异（简称“位置差异”，下同），给出这一个指标的原因很明显，它直接描述了算法是否跟到了目标；

第二，每一帧图片中算法给出的框选区域大小与标准结果在图片中的大小差异（简称“尺度差异”，下同），给出这个指标的原因在于我们需要衡量算法是否将不该跟踪的部分视为目标以及是否将应该跟踪的目标部分未计入框内。

4.2.2 各指标优先度

第一，因为我们的算法要应用到机器人上面，因此，跟踪算法必须能够做到实时处理，也就是说其帧率要足够的高。这一点使我们在选择算法的时候就需要考虑的。对于这一点 KCF 算法表现优异，我们可以不用担心。所以 ，帧率不是我们需要优化的目标，至少不是第一目标（因为 KCF 已经在这方面有很好表现）。

第二，对于精度，前以提及，我们要做两方面的考虑，即位置差异和尺度差异（详见 4.2.1 节）。在这两者之间我们认为位置差异更加重要。原因在于一旦

位置有偏差，框内的对象必然不是目标。只有在位置正确的前提之下，尺度正确才有意义。

综上所述，我们优化的三个指标有如下优先度：

帧率>位置差异>尺度差异

而我们真真需要优化的两个目标就是：

位置差异>尺度差异

4.3 优化指标的量化度量

4.3.1 位置差异的量化度量

对于位置差异，我们在描述的时候采用测试框和标准框（见 4.2.1 节）中心点坐标的距离来描述。设标准框和测试框中心点的坐标分别为(x_0,y_0)和(x,y)，那么我们的位置差异就可以表示为：

$$d=\sqrt{(x-x_0)^2+(y-y_0)^2}$$

d 值越小说明算法给出的结果越好。

==========注====解==========

在这个问题的解决过程中，我们最先提出了另外一个优化量，即比较测试框与标准框左上角的坐标之间的距离。但最后发现，即使两者相差为零，也会因为尺度差异的影响使得位置差异无法正确反映。

也就是说，这个量会与尺度差异相耦合。基于此，我们认为中心点坐标之间的距离可以更好刻画位置差异，因为此时位置差异不再因为尺度改变而产生变化。

==========

4.3.2 尺度差异的量化度量

记测试框的宽高分别为w,h，标准框的宽高分别为w_0,h_0，那么尺度差异s就可以简单地定义为：$s=\sqrt{wb/w_0b_0}$。最优结果为$s=1$也就是测设结果与标准结果框的大小一样。

说明：对于位置差异和尺度差异的定量描述可以定义很多数学量对其刻画，但关键在于要反映出我们感兴趣的方面，同时也要将实际应用背景考虑进去。

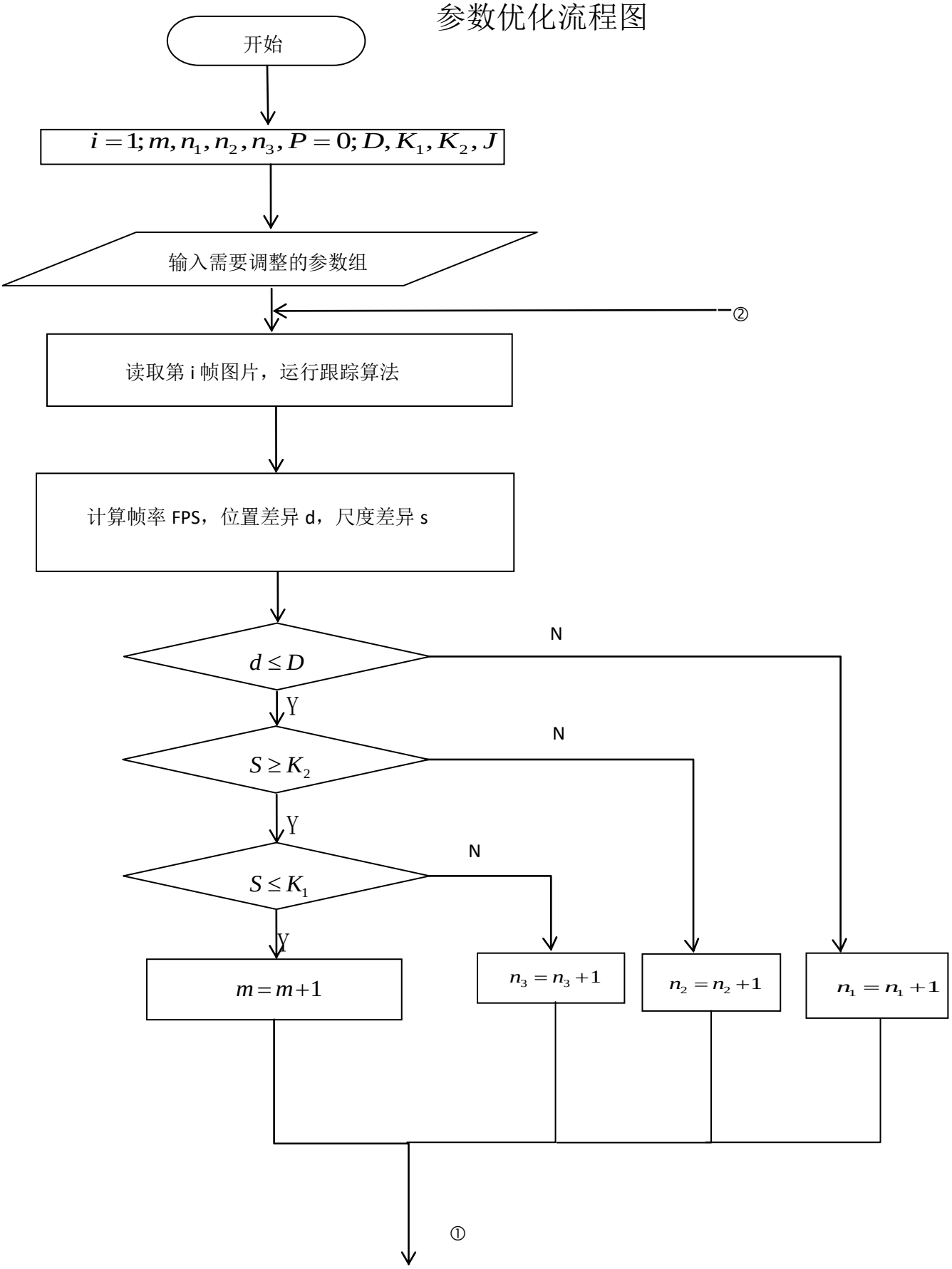
4.4 优化流程

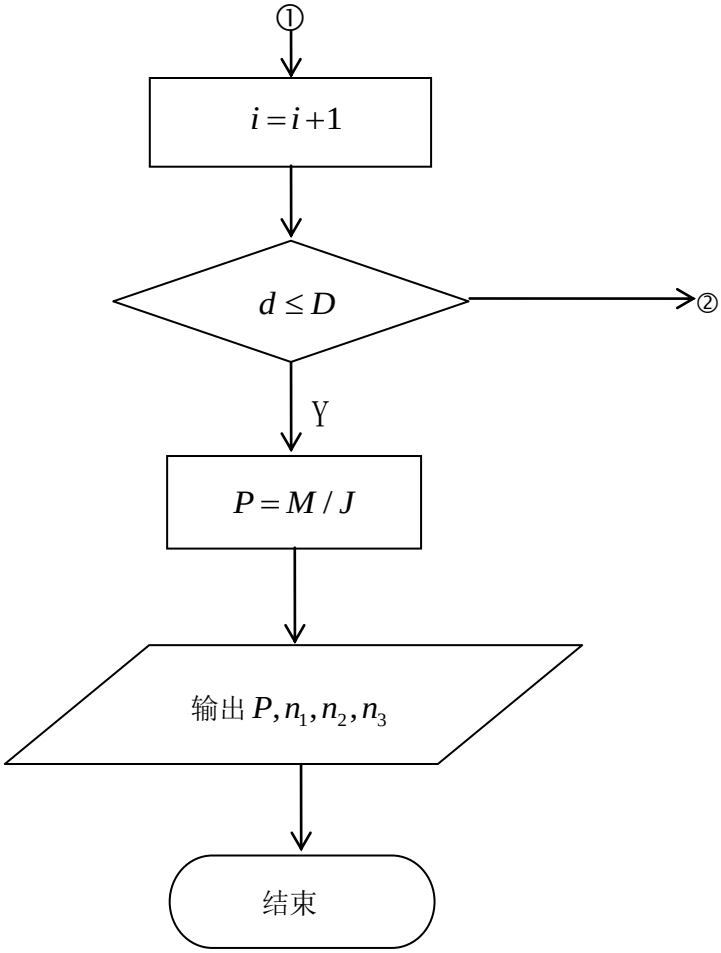
在实际优化过程中，我们先要调整参数，运行算法，然后计算我们上面设计出的目标量d,s还有帧率。

在得到目标量d,s还有帧率以后，我们需要设定位置差异和尺度差异的阈值，位置差异因为恒大于零，所以只有一个上界（D），大于此值我们认为跟踪失败，尺度差异有下界K_1和上界K_2，过小和过大均认为跟踪失败。**在优先保证位置差异得到保证的情况下，之要尺度差异不是很过分，我们则认为跟踪成功**。最后将跟踪成功的帧数与总帧数做除法，得到精度值。在整个优化过程中，我们对跟踪失败的情况作了归因处理。即就是对于跟踪失败的图片分位置差异（n_1）、尺度差异过大（n_2）、尺度差异过小（n_3）分别计数，这样做的目的是便于我们有针对性的进行优化。

我们在后面附有跟踪算法的优化流程。

参数优化流程图


开始
$i=1;m,n_1,n_2,n_3,P=0;D,K_1,K_2,J$
输入需要调整的参数组
②
读取第 i 帧图片，运行跟踪算法
计算帧率 FPS，位置差异 d，尺度差异 s
$d \le D$
N
Y
$S \ge K_2$
N
Y
$S \le K_1$
N
Y
$m=m+1$
$n_3=n_3+1$
$n_2=n_2+1$
$n_1=n_1+1$
①


①
$i=i+1$
$d \le D$
②
Y
$P=M/J$
输出 P, n_1, n_2, n_3
结束

4.5 参数优化的讨论

在讨论问题之前先阐明一个概念。我们把能够实现同一个目标的多种算法称为**算法体系。**比如所有能够实现计算机视觉跟踪的算法我们就可以将其称为算法体系。对于某个算法体系，因为体系内的算法都可以实现相同的目标，所以我们可以以此建立起关于该体系内算法的统一评判标准。这样的标准在实际应用过程中有以下两方面的作用：

第一，对于不同的算法，我们可以以此标准评判它们的综合表现；

第二，对于同一个算法，我们可以以此标准对其进行参数优化。

上面两条，我们更加注重的是第二条。原因在于：第一，选择某一个算法往往取决于其他更多的因素（版权，硬件平台兼容性等等）。而一旦选定了其中的一个算法，我们就有必要对其进行精雕细琢（参数优化），而这一步工作就是第二条的内容；第二，参数优化本身是一项工作量很大的工作，如果在选择某个算法的问题上用这种方法未免得不偿失。

需要说明的是，上面所说的优化标准在制定的时候，必须考虑到实际应用背景，否则这样的标准并不能很好的引导我们挖掘算法的潜力。

总而言之，一个算法的适用大致经历下面几个过程：

第一，实际问题的提出。此时需要我们提出所面临的问题，应该尽可能的细致、精准地描述我们需要解决的一系列问题；

第二，针对所提出的问题，找出一系列可以解决的算法，列成清单并注明每一种算法所能解决的问题，以及其自身的缺陷；

第三，对我们的问题进行优先度排序，然后以此找出能够解决最重要问题的算法（具体操作应该有微调，受具体情况影响），以此类推找到一系列备选的解决方案；

第四，结合版权，硬件平台，费用支出等因素作进一步选择，确定要使用的算法；

第五，针对所采用的算法进行参数优化。

注：上面所说的算法，有时指的是程序代码，在这里不加区分。

附录

A、循环矩阵

从形式上看，循环矩阵的各行由一个行向量以及这个行向量的循环移位生成。我们可以从不同的角度对循环矩阵进行定义，从而使得循环矩阵的良好性质得到更好的发挥。

A.1 循环矩阵的定义

A.1.1

我们称如下形式的矩阵为循环矩阵：

$$X = C(\mathbf{x}) = \begin{bmatrix} x_0 & x_1 & x_2 & \cdots & x_{n-1} \\ x_{n-1} & x_0 & x_1 & \cdots & x_{n-2} \\ x_{n-1} & x_n & x_0 & \cdots & x_{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1 & x_2 & x_3 & \cdots & x_0 \end{bmatrix}$$

其中，循环矩阵的第一行为 $\mathbf{x} = [x_0, x_1, \cdots, x_{n-1}]$，我们称其为循环矩阵的生成向量，其后每行，只需依次对其做循环移即可得到。也就是说，循环矩阵可以由其生成向量完全确定。

仔细观察就会发现，我们也可以用第一列来做循环移位操作生成循环阵，但为了方便利用离散傅里叶变换，我们只认可用行的循环移位来定义的方式。

循环矩阵的这种定义是它的诸多定义中最直观最容易为人们所接受的一种定义。但是这种定义的直观性也直接限制了它的其他数学性质的表达。比如，这个定义中没有提供一种可以用作描述循环移位的数学语言。为此我们引入如下定义。

A.1.2

正如上面所叙述的，我们应该找一种数学语言来描述循环移位，这样我们就可以用纯粹的数学语言来描述循环矩阵，而不是叙述性的语句。

在对循环矩阵做第二中定义之前，我们需要先引入一个概念，定义矩阵

$$P=\begin{bmatrix}0&0&0&\cdots&1\\1&0&0&\cdots&0\\0&1&0&\cdots&0\\\vdots&\vdots&\ddots&\ddots&\vdots\\0&0&\cdots&1&0\end{bmatrix}$$

我们将矩阵 P 称为置换矩阵(permutation matrix)或者循环移位算子(cyclic shift operator)。我们可以作如下的简单运算看见它的功能：

$$P\mathbf{x}^T=\begin{pmatrix}0&0&0&\cdots&1\\1&0&0&\cdots&0\\0&1&0&\cdots&0\\\vdots&\vdots&\vdots&\ddots&\vdots\\0&0&\cdots&1&0\end{pmatrix}\begin{pmatrix}x_0\\x_1\\x_2\\\vdots\\x_{n-1}\end{pmatrix}=\begin{pmatrix}x_{n-1}\\x_0\\x_1\\\vdots\\x_{n-2}\end{pmatrix}$$

从上面的运算可以看出，矩阵 P 可以对向量进行循环移位操作，

即P矩阵将序列$[x_0,x_1,x_2,\cdots,x_{n-1}]^T$转换为$[x_{n-1},x_0,x_1,\cdots,x_{n-2}]^T$

因为矩阵 P 的作用是循环移位，所以多次移位就可以用矩阵 P 的幂次来表示，从而我们可以用如下方式定义循环矩阵：

$$C(\mathbf{x})=\begin{bmatrix}(P^0\mathbf{x}^T)^T\\(P^1\mathbf{x}^T)^T\\\vdots\\(P^{n-1}\mathbf{x}^T)^T\end{bmatrix}$$

到这里我们就可以直接用纯粹的数学语言来描述循环矩阵了！

我们虽然解决了用数学语言来描述循环矩阵的问题，这对于我们做与循环矩阵相关的运算是非常有益的，但是这样的定义并不利于我们去判定一个矩阵是否是循环矩阵。所以，我们引入第三种定义。

A. 1. 3

设有向量$\mathbf{x}=[x_0,x_1,\cdots,x_{n-1}]$若对于矩阵 X 有 $X_{i,j}=x_{(j-i)\bmod n}$，那么我们就称 X 为由向量 x 生成的循环矩阵。（R. M. Gray, Toeplitz and Circulant Matrices A Review. Now.pdf）

需要说明的是，取余数运算(mod)在这里要拓展一下，即当(j-i)为负数的时候需要将其变为 n+(j-i)然后再取余，关于这一点可以用取余运算的周期性理解，它

们与循环移位的周期性相对应。

这个定义是循环矩阵的定义中最抽象的一个，它很不直观，但是利用这个定义却可以很容易判定一个矩阵是否是循环矩阵。

A.2 循环矩阵的性质

A.2.1 **循环矩阵对加法和数乘封闭**，即就是：

若 A，B 均为 n 循环矩阵,则对于任意常数 a 和 b,（aA+bB)也是循环矩阵。

A.2.2 **所有的循环矩阵都可以酉对角化,并且对角阵的元素（X 的特征值）就是生成向量的傅里叶变换，该酉矩阵就是傅里叶变换矩阵的单位化。即：**

若 X 为 n 阶循环矩阵，那么必有

$$F=\frac{1}{\sqrt{n}}\begin{pmatrix}1 & 1 & 1 & \cdots & 1\\ 1 & \omega & \omega^2 & \cdots & \omega^{n-1}\\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)(n-1)}\end{pmatrix}_{n\times n}$$ 使得：

$$X=Fdiag(\hat{\mathbf{x}})F^T$$

其中，ω 是 1 的 n 次单位根（即 $\omega^n-1=0$），$\hat{\mathbf{x}}$ 是矩阵 X 的生成向量 $\mathbf{x}$ 的傅里叶变换（即 $\hat{\mathbf{x}}=\sqrt{n}F\mathbf{x}$）。

这一性质是循环矩阵非常重要的一条性质，正是由于这条性质的存在，使得 KCF 有很快的速度。

A.2.3 **循环矩阵的转置矩阵还是循环矩阵**。

这一条性质我们可以通过循环矩阵第一个定义(A.1.1)得到。

B、离散傅里叶变换（DFT）

B.1 定义

设有序列 $\mathbf{z}=[z_0,z_1,\cdots,z_{n-1}]^T$，ω 是 1 的 n 次单位根（$\omega^n-1=0$），于是 $\mathbf{z}$ 的 DFT 为 $\hat{\mathbf{z}}^T=[\hat{z}_0,\hat{z}_1,\cdots,\hat{z}_{n-1}]^T$，其中 $\hat{z}_i=\sum_{k=0}^{n-1}z_k\omega^{ik}$。写为矩阵形式，即就是：

$\hat{\mathbf{z}}=\sqrt{n}F\mathbf{z}$ 记为 $\wp(\mathbf{z})$（F 矩阵的定义参考 A.2.2）

B.2 性质

B.2.1 线性性

所谓线性性指的是向量的 DFT 与向量的线性组合两种运算可以交换次序。即就是：$\wp(a\mathbf{x}+b\mathbf{y})=a\,\wp(\mathbf{x})+b\,\wp(\mathbf{y})$。

这条性质可以由 DFT 的矩阵表达式以及矩阵运算的线性性质得到。

B.2.2 循环逆序性

在介绍本条性质之前先介绍循环逆序的概念：

向量 $\mathbf{x}=[x_0,x_1,\cdots,x_{n-1}]$ 的循环逆序为 $\bar{\mathbf{x}}=[x_0,x_{n-1},\cdots,x_1]$。

注意这里逆序与我们传统逆序概念的不同之处。

向量 z 的逆序 $\bar{\mathbf{z}}$ 的 DFT 等于 z 的 DFT 的共轭。

即：$\wp(\bar{\mathbf{z}})=\wp^*(\mathbf{z})$，“*”表示取共轭。

本条性质可由 n 次单位根 $\omega^k=\overline{\omega}^{n-k}$ 得到（将其带入 DFT 定义式中即可得到）。

C、（循环）卷积

C.1 定义

设两个 n 维向量 $\mathbf{x}=[x_0,x_1,\cdots,x_{n-1}]$，$\mathbf{y}=[y_0,y_1,\cdots,y_{n-1}]$，它们的卷积为 $\mathbf{z}=\mathbf{x}*\mathbf{y}$，则 z 的第 i 个分量 $z_i=\sum_{k=0}^{n-1}x_{(i-k)\bmod n}y_k$。

C.2 卷积与循环矩阵的联系（henriques_phd.pdf P37）

记两个 n 维向量 $\mathbf{x}=[x_0,x_1,\cdots,x_{n-1}]$，$\mathbf{y}=[y_0,y_1,\cdots,y_{n-1}]$，它们的卷积 $\mathbf{z}=\mathbf{x}*\mathbf{y}=C^T(\mathbf{x})\mathbf{z}$。

==========解=====释==========

关于这个性质，只需要将 $C^T(\mathbf{x})\mathbf{z}$ 展开，然后与卷积定义式作比较即可得到，出于直观性，在这里我们给出一个小例子以供参考：

设$x(n)=\{3,2,1,4\}, h(n)=\{1,1,2,0\}$,求$y(n)=x(n)\otimes h(n), N=4$

解：N = 4时，

$$h(\underline{n})=\{1,1,2,0\}, \quad x(0)h(\underline{n})=\{3,3,6,0\}$$
$$h(\underline{n-1})=\{0,1,1,2\}, \quad x(1)h(\underline{n-1})=\{0,2,2,4\}$$
$$h(\underline{n-2})=\{2,0,1,1\}, \quad x(2)h(\underline{n-2})=\{2,0,1,1\}$$
$$h(n-3)=\{1,2,0,1\}, \quad x(3)h(n-3)=\{4,8,0,4\}$$

将结果累加的$y(n)=\{9,13,9,9\}$,以上过程用竖式计算如下：

$$\begin{array}{cccc} 1 & 1 & 2 & 0 \\ 3 & 2 & 1 & 4 \\ \hline 3 & 3 & 6 & 0 \\ 0 & 2 & 2 & 4 \\ 2 & 0 & 1 & 1 \\ 4 & 8 & 0 & 4 \\ \hline 9 & 13 & 9 & 9 \end{array}$$

表示为矩阵乘法 $\rightarrow$

$$\begin{bmatrix} 1 & 0 & 2 & 1 \\ 1 & 1 & 0 & 2 \\ 2 & 1 & 1 & 0 \\ 0 & 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \\ 1 \\ 4 \end{bmatrix} = \begin{bmatrix} 9 \\ 13 \\ 9 \\ 9 \end{bmatrix}$$

$$\mathbf{h}\otimes\mathbf{x}=C(\bar{\mathbf{h}})\mathbf{x}=C^T(\mathbf{h})\mathbf{x}=\mathbf{y}$$

（$\bar{\mathbf{h}}$ 表示循环逆序。）

==========