Robust Estimation of 3D Human Poses from a Single Image
论文复现工作总结

张淦淦 MF20230144

1 人体姿态识别研究前沿

针对单人人体姿态识别主要的数据库有两种类型

- 第一种类型是RGB图像与对应的2D关键点标注,如MPI数据库,具有大量自然场景下人体的姿态标注.
- 第二种类型是RGB图像与对应的2D关节点标注和通过运动传感器设备得到对应3D数据,如HumanEVA,Human3.6m等,这种类型数据库3D姿态主要在实验室环境下标注.

从RGB中识别出2D人体关节点数据常用的网络结构有[1,2]

论文 [3]利用第二种类型的数据集,以2D关节点数据作为输入,3D姿态数据作为标记,视人体姿态2D到3D转换为一个回归问题,成功训练了一个简单的神经网络完成任务.如果以stacked hourglass网络 [2]从RGB中提取的2D关节点作为输入,可以通过微调得到SOTA效果,若以真实2D作为输入时,效果可以远远超过SOTA.

自从 [3]提出后,由2D关节点提升到3D关节点的深度学习方法的研究从两个角度进行.

改变模型结构 为了提升网络的运行或学习速度,减少参数量,核心思路是通过利用改变网络结构或者特征表达的方式起到添加先验知识的目的.

[4]是在 [3]网络的基础上添加了一层Bidirect RNN网络,Bidirect RNN可以作为对原有网络一种约束,即关节点只与相邻的关节点有关,作者定义了一系列约束(grammar),然后用网络结构表达了这种约束,起到了更好的效果.这种思想可以理解为利用RNN实现图神经网络的一个特例.

[5]和 [6]都是利用图神经网络来解决输入2D姿态输出3D姿态的问题,前者除了应用图神经网络架构,还利用了原始图像信息.以上两者对比基准 [3],在精度一样的条件下,网络参数量大大减小. [5]使用了更多的技巧让总体效果更好,达到了SOTA,而 [6]只是扩展了图卷积神经网络,有很好的启发意义.

提升模型泛化能力 加强深度学习泛化能力的通用方法有

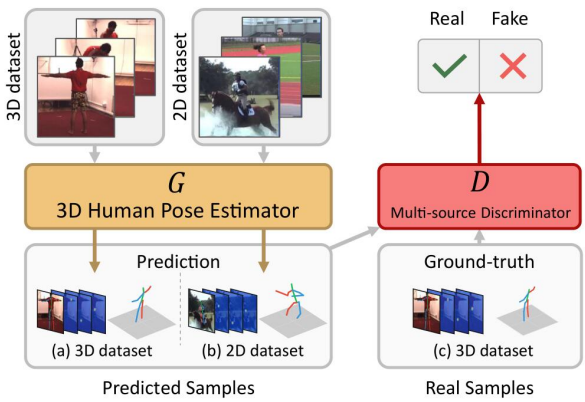
- 为损失函数添加正则项
- 集成学习(个数5-10个)
- Dropout,在训练中广泛使用
- 增强数据集,这个由3D骨架,通过设置相机参数大量生成对应的2D图像,此外,还可以通过添加噪声等方法进一步扩大数据规模.

针对3D姿态识别问题,目前现存数据集中含有的3D姿态数据,主要从实验室室内采集,远远不能覆盖日常生活中的人体的姿态.3D数据的采集成本较大,故不能大规模采集,相对而言,2D姿态数据采集成本较小,目前数据集覆盖大量自然场景下人的姿态.

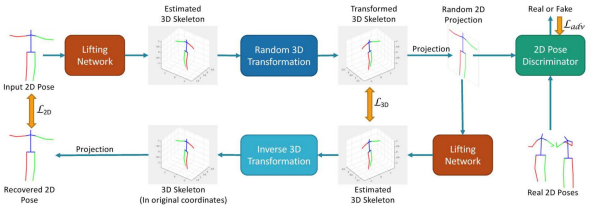
半监督学习方法同时采用2D数据集(RGB图像和2D标注)和3D数据集(RGB图像,2D标注,3D数据)进行训练,以期望利用更多数据提升模型精度的同时也提升自然场景下泛化能力.

论文 [7]中,通过精巧定义一种能同时适用于2D数据集和3D数据集的损失函数,用MPI数据集和Human3.6m的数据端到端训练Stack Houglass网络(RGB到2D)和深度回归网络(2D到3D),并且还保留了原始RGB中的特征向量与2D数据同时作为深度回归网络的输入.

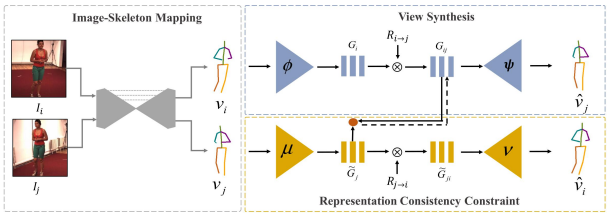
利用对抗学习方式融合只有2D标注的数据集和含有3D标注的数据集是一种十分自然的方式. [8]



无监督学习方法,如 [9]中纯以2D数据为输入,以下图直观显示了其思路



对同一个对象得到多个不同位置相机下对应图像比获得3D姿态信息成本相对低廉.所以, [10, 11]都是利用多个视角的图片利用编码器学习描述3D信息的特征向量,然后用神经网络对编码器提取出来的特征向量进行学习.这样处理,精度相对于强监督学习有所下降,但泛化能力更强



2 论文内容

2.1 概要

单幅图像3D人体姿态估计,可以分为两个阶段进行.第一个是由图像得到2D关键点信息,可分为单人2D关键点检测,多人2D关键点检测(单人2D关键点检测研究已经比较成熟,相关方法如stacked hourglass(2017)),第二个阶段是阶段是从已知2D信息估计出3D姿势信息.本文讨论的是已知单人2D关键点信息估计出3D姿势信息.

2.2 基本模型

假设n表示关键点的数量,2D姿态与3D姿态分别用$x \in R^{2n}$和$y \in R^{3n}$表示,$M_0 = \begin{pmatrix} m_1^T \\ m_2^T \end{pmatrix} \in R^{2\times 3}$表示弱透视模型相机矩阵

$$M = I_n \otimes M_0 \in R^{2n\times 3n}$$

$$x = My$$

对于世界坐标系中的y,先将其变换到以物体为基准的坐标系,即以左肩到右肩为x轴,以两肩中点到腰为y轴,z轴是x与y的叉乘.然后以各个关节点各轴上点取平均得到$\mu \in R^3$,以其为原点.

人体姿态在以往的研究中,很适合用在稀疏空间中表示 [12],设$B = b_1, \ldots, B_k \in R^{3n\times k}$是通过字典学习的方法学习到的字典 [13](实际中用spams库实现,k是自己设定的字典长度,本次实验设计为200),$\alpha \in R^{k\times 1}$是对应的稀疏向量.下式中μ为0

$$y = B\alpha + \mu$$

所以,根据2D关键点估计3D姿态可以转化为,一个优化问题

$$\min_{\alpha} ||x - M(B\alpha + \mu)||_1 + \theta||\alpha||$$

加入约束,设肢体各个部分的长度分别为L_i,则有

$$||C_i y||_2^2 = L_i$$

$$C_i = E_{i1} - E_{i2}$$

$$E_j = [0, \ldots, I, \ldots, 0] \in R^{3\times 3n}$$

原问题转换为优化问题

$$\begin{aligned} \min_{\alpha} \quad & ||x - M(B\alpha + \mu)||_1 + \theta||\alpha|| \\ s.t. \quad & ||C_i(B\alpha + \mu)||_2^2 = L_i \end{aligned} \tag{1}$$

实际中,相机M_0参数不可知,所以,采用的式循环迭代的方式,先估计相机的参数,然后根据估计得到的参数计算上述优化问题,得到姿态y,再根据姿态y求下述优化问题的解以得到相机参数

$$\begin{aligned} \min_{m_1, m_2} \quad & ||X - \begin{pmatrix} m_1^T \\ m_2^T \end{pmatrix} Y||_1 + \theta||\alpha|| \\ s.t. \quad & m_1^T m_2 = 0 \end{aligned} \tag{2}$$

2.3 小结

1. 对y进行坐标变换,归一化.通过字典学习得到B
2. 设定y_0的初始化值,
3. 将y_k带入(2)求得M_k
4. 将M_k带入(1),求得y_{k+1}
5. 3,4循环直到$||M_k - M_{k-1}||_2 < \sigma_1$和$||y_k - y_{k-1}||_2 < \sigma_1$变动

3 论文中求解优化问题所使用的数值算法

3.1 ADMM,LADM与LADMPSAP

3.1.1 ADMM

参考自 [14],ADMM算法可以解决以下形式的优化问题

$$\begin{aligned} min \quad & f(x) + g(z) \\ s.t. \quad & Ax + Bz = c \end{aligned}$$

其中f和g都是凸函数

其增强拉格朗日方程

$$L_\rho(x,z,y) = f(x)+g(z)+y^T(Ax+Bz-c)+\frac{\rho}{2}||Ax+Bz-c||_2^2$$

ADMM算法描述 ADMM算法迭代方程

$$\begin{aligned} x^{k+1} &= \underset{x}{argmin}\, L_\rho(x, z^k, y^k) \\ z^{k+1} &= \underset{z}{argmin}\, L_\rho(x^{k+1}, z, y^k) \\ y^{k+1} &= y^k + \rho(Ax^{k+1} + Bz^{k+1} - c) \end{aligned}$$

另外用一种Scaled Form,由

$$y^T r + \frac{\rho}{2}||r||_2^2 = \frac{\rho}{2}||r + \frac{1}{\rho}y||_2^2 - \frac{1}{2\rho}||y||_2^2$$

$$\begin{aligned} x^{k+1} &= \underset{x}{argmin}\left(f(x) + \frac{\rho}{2}||Ax + Bz^k - c + u^k||_2^2\right) \\ z^{k+1} &= \underset{z}{argmin}\left(g(z) + \frac{\rho}{2}||Ax^{k+1} + Bz - c + u^k||_2^2\right) \\ u^{k+1} &= u^k + Ax^{k+1} + Bz^{k+1} - c \end{aligned} \tag{3}$$

或

$$\begin{aligned} x^{k+1} &= \underset{x}{argmin}\left(f(x) + \frac{\beta}{2}||Ax + Bz^k - c + \lambda^k/\beta||_2^2\right) \\ z^{k+1} &= \underset{z}{argmin}\left(g(z) + \frac{\beta}{2}||Ax^{k+1} + Bz - c + \lambda^k/\beta||_2^2\right) \\ \lambda^{k+1} &= \lambda_k + \beta[Ax^{k+1} + Bz^{k+1} - c] \end{aligned} \tag{4}$$

将不等式约束转换为ADMM可解形式 具体参见**??**第5章,广泛存在的约束凸优化问题可以写作

$$min \quad f(x)\, s.t. \qquad x \in C$$

其中f是凸函数,C是凸集

$$\begin{aligned} min \quad & f(x) + g(z) \\ s.t. \quad & x - z = 0 \end{aligned}$$

其中g为指示函数,起到限制x在C中的作用

$$\begin{aligned} x^{k+1} &= \underset{x}{argmin}\left(f(x) + \frac{\rho}{2}||Ax + Bz^k - c + u^k||_2^2\right) \\ z^{k+1} &= \prod_C (x^{k+1} + u^{k+1}) \\ u^{k+1} &= u^k + x^{k+1} + z^{k+1} \end{aligned}$$

z的更新是通过$\prod_C$,是欧几里德投射到C的操作.

欧几里德投射是近端操作(proximal operator)的一个特例,一个点$x_0 \in R^n$在一个集合$S \in R^n$上的欧式投影式满足$x_0 \in R^n$到该点距离是到集合的最短欧式距离,即

$$\begin{aligned} \underset{x}{min} \quad & ||x - x_0||^2 \\ s.t. \quad & x \in S \end{aligned}$$

3.1.2 LADMAP

见 [15],是为了加速ADM算法而提出,针对LRR问题的实验结果如下

Size (s, p, d, $\tilde{r}$)	Method	Time	Iter.	$\frac{\|\hat{\mathbf{Z}}-\mathbf{Z}_0\|}{\|\mathbf{Z}_0\|}$	$\frac{\|\hat{\mathbf{E}}-\mathbf{E}_0\|}{\|\mathbf{E}_0\|}$	Acc.
(10, 20,200, 5)	APG	0.0332	110	2.2079	1.5096	81.5
	ADM	0.0529	176	0.5491	0.5093	**90.0**
	LADM	0.0603	194	**0.5480**	**0.5024**	**90.0**
	LADMAP	0.0145	**46**	**0.5480**	**0.5024**	**90.0**
	LADMAP(A)	**0.0010**	**46**	**0.5480**	**0.5024**	**90.0**
(15, 20,300, 5)	APG	0.0869	106	2.4824	1.0341	80.0
	ADM	0.1526	185	0.6519	0.4078	83.7
	LADM	0.2943	363	**0.6518**	**0.4076**	**86.7**
	LADMAP	0.0336	**41**	**0.6518**	**0.4076**	**86.7**
	LADMAP(A)	**0.0015**	**41**	**0.6518**	**0.4076**	**86.7**
(20, 25, 500, 5)	APG	1.8837	117	2.8905	2.4017	72.4
	ADM	3.7139	225	1.1191	1.0170	80.0
	LADM	8.1574	508	**0.6379**	**0.4268**	80.0
	LADMAP	0.7762	**40**	**0.6379**	**0.4268**	**84.6**
	LADMAP(A)	**0.0053**	**40**	**0.6379**	**0.4268**	**84.6**
(30, 30, 900, 5)	APG	6.1252	116	3.0667	0.9199	69.4
	ADM	11.7185	220	0.6865	0.4866	**76.0**
	LADM	N.A.	N.A.	N.A.	N.A.	N.A.
	LADMAP	2.3891	**44**	**0.6864**	**0.4294**	**80.1**
	LADMAP(A)	**0.0058**	**44**	**0.6864**	**0.4294**	**80.1**

可见,在进度下降可接受的范围内,算法速度大大提升

LADM linearzed ADM,是通过将(4)中二次项线性化做近似,得到递推公式

$$\begin{aligned} x^{k+1} &= \underset{x}{argmin}\, f(x) + \frac{\beta\eta_A}{2}||x - x^k + \\ &\quad A^*(\lambda^k + \beta(Ax^k + Bz^k - c))/(\beta\eta_A)||_2^2 \\ z^{k+1} &= \underset{z}{argmin}\, g(x) + \frac{\beta\eta_B}{2}||z - z^k + \\ &\quad B^*(\lambda^k + \beta(Ax^{k+1} + Bz^k - c))/(\beta\eta_B)||_2^2 \\ \lambda^{k+1} &= \lambda_k + \beta[Ax^{k+1} + Bz^{k+1} - c] \end{aligned} \tag{5}$$

A^*是A的伴随矩阵.

LADMAP 加入自适应惩罚项,以加速收敛

$$\beta_{k+1} = min(\beta_{max}, \rho\beta_k)$$

$$\rho = \begin{cases} \rho_0, if\, \beta_k max(\sqrt{\eta_A}||x^{k+1} - x^k||, \sqrt{\eta_B}||z^{k+1} - z^k||)/||c|| < \varepsilon \\ 1, otherwise \end{cases}$$

3.1.3 LADMPSAP

上文描述的LADMAP可以解决tow blocks形式问题(x,z两个变量),LADMPSAP是LADMAP的扩展,用于解决多个block的问题

$$\begin{aligned} \underset{x_1,\ldots,x_n}{min} \quad & \sum_{i=1}^n f_i(x_i) \\ s.t. \quad & \sum_{i=1}^n A_i x_i = b \end{aligned}$$

LADMAP可以解决i=1,2形式的问题,重写为

$$
\begin{aligned}
x_1^{k+1} =& \underset{x_1}{argmin}\, f_1(x_1) + \frac{\beta^k\eta_1}{2}||x_1 - x_1^k + \\
& A^*(\widetilde{\lambda}_1^k + \beta(A_1x_1^k + A_2x_2^k - c))/(\beta^k\eta_1)||_2^2 \\
x_2^{k+1} =& \underset{x_2}{argmin}\, f_2(x_2) + \frac{\beta^k\eta_2}{2}||x_2 - x_2^k + \\
& A^*(\widetilde{\lambda}_2^k + \beta(A_1x_1^{k+1} + A_2x_2^k - c))/(\beta^k\eta_2)||_2^2 \\
\lambda^{k+1} =& \lambda^k + \beta^k(\sum_{i=1}^{2} A_ix_i^{k+1} - b) \\
\widetilde{\lambda}_1^k =& \lambda^k + \beta^k(A_1x_1^k + A_2x_2^k - b) \\
\widetilde{\lambda}_2^k =& \lambda^k + \beta^k(A_1x_1^{k+1} + A_2x_2^k - b) \\
\beta_{k+1} =& min(\beta_{max}, \rho\beta_k)
\end{aligned}
\tag{6}
$$

对于multi block问题,LADMPSAP迭代方程为

$$
\begin{aligned}
x_i^{k+1} =& \underset{x_i}{argmin}\, f_i(x_i) + \frac{\beta^k\eta_i}{2}||x_i - x_i^k + \\
& A^*(\widetilde{\lambda}^k + \beta(\sum_{m=1}^{i-1} A_mx_m^{k+1} + \sum_{m=i+1}^{n} A_mx_m^{k+1} - c))/(\beta^k\eta_i)||_2^2 \\
\lambda^{k+1} =& \lambda^k + \beta^k(\sum_{i=1}^{n} A_ix_i^{k+1} - b) \\
\beta_{k+1} =& min(\beta_{max}, \rho\beta_k) \\
\widetilde{\lambda}^k =& \lambda^k + \beta^k(\sum_{i=1}^{n} A_ix_i^{k+1} - b)
\end{aligned}
$$

$$
\rho = \begin{cases} \rho_0, if\beta_k max(\{\sqrt{\eta_i}||x^{k+1} - x^k||)/||c|| < \varepsilon_2\} \\ 1, otherwise \end{cases}
\tag{7}
$$

3.2 用CVXPY解决优化问题

优化问题的求解往往需要将一个特定问题转换为数值求解器(solver)要求的固定格式进行求解,cvxpy是一个可以将一定满足规则的优化模型,自动转换为求解器需要的固定格式,避免手工转换.

cvxpy用DCP(Disciplined convex programming)分析器保证特定问题是凸问题.cvxpy可以解决满足DCP规则的所有问题,以下解释DCP规则

DCP规则 要求目标函数满足形式

- Minimize(convex)
- Maximize(concave)

The only valid constraints under the DCP rules are

- affine == affine
- convex <= concave
- concave >= convex

Curvature rules

DCP analysis is based on applying a general composition theorem from convex analysis to each (sub)expression.

$f(\text{expr}_1, \text{expr}_2, \ldots, \text{expr}_n)$ is convex if f is a convex function and for each expr_i one of the following conditions holds:

- f is increasing in argument i and expr_i is convex.
- f is decreasing in argument i and expr_i is concave.
- expr_i is affine or constant.

$f(\text{expr}_1, \text{expr}_2, \ldots, \text{expr}_n)$ is concave if f is a concave function and for each expr_i one of the following conditions holds:

- f is increasing in argument i and expr_i is concave.
- f is decreasing in argument i and expr_i is convex.
- expr_i is affine or constant.

$f(\text{expr}_1, \text{expr}_2, \ldots, \text{expr}_n)$ is affine if f is an affine function and each expr_i is affine.

If none of the three rules apply, the expression $f(\text{expr}_1, \text{expr}_2, \ldots, \text{expr}_n)$ is marked as having unknown curvature.

Curvature

Each (sub)expression is flagged as one of the following curvatures (with respect to its variables)

Curvature	Meaning
constant	$f(x)$ independent of x
affine	$f(\theta x + (1-\theta)y) = \theta f(x) + (1-\theta)f(y),\ \forall x,\ y,\ \theta \in [0,1]$
convex	$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y),\ \forall x,\ y,\ \theta \in [0,1]$
concave	$f(\theta x + (1-\theta)y) \geq \theta f(x) + (1-\theta)f(y),\ \forall x,\ y,\ \theta \in [0,1]$
unknown	DCP analysis cannot determine the curvature

对于原问题

$$
\begin{aligned}
\underset{\alpha}{min} \quad & ||x - M(B\alpha + \mu)||_1 + \theta||\alpha|| \\
s.t. \quad & ||C_i(B\alpha + \mu)||_2^2 = L_i
\end{aligned}
\tag{8}
$$

由DCP规则分析,目标函数符合DCP规则(L_1范数为凸函数,$x - M(B\alpha + \mu)$,α是仿射函数),但是约束显然不是仿射函数,所以不符合DCP规则,不能用cvxpy求解.即不能转换为现存的优化解析器需要的模型格式

4 论文中优化问题具体求解

4.1 姿态识别求解

论文第二部分重点讨论了如何求解(1)和(2),由于使用CVX求解的DCP规则,不能转化为常用的数值解析器需要的优化问题格式(见3.2的讨论),此外,就算CVX使用的解算器多基于内点法,效率低下(如解决最小化$f(X) = ||X||_1$,每个迭代的复杂度为$O(q^6)$,其中$q \times q$为矩阵的大小).转换原优化问题的格式,首先引入β, γ,将(1)转换为,方便借用ADMM方法求解([16])

$$
\begin{aligned}
\underset{\alpha,\beta,\gamma}{min} \quad & ||\gamma||_1 + \theta||\beta|| \\
s.t. \quad & \gamma = x - M(B\alpha + \mu) \\
& \alpha = \beta \\
& ||C_i(B\alpha + \mu)||_2^2 = L_i, i = 1, 2, \ldots, m
\end{aligned}
\tag{9}
$$

其增强拉格朗日方程为

$$L_1(\alpha,\beta,\gamma,\lambda_1,\lambda_2,\eta) = ||\gamma||_1 + \theta||\beta||_1 + \lambda_1^T[\gamma - x + M(B\alpha+\mu)] + \lambda_2^T(\alpha-\beta) + \frac{\eta}{2}[||\gamma - x + M(B\alpha+\mu)||^2 + ||\alpha-\beta||^2]$$

即

$$\begin{aligned} &\min_{\alpha,\beta,\gamma,\lambda_1,\lambda_2} \quad ||\gamma||_1 + \theta||\beta||_1 + \lambda_1^T[\gamma - x + M(B\alpha+\mu)] \\ &+ \lambda_2^T(\alpha-\beta) + \frac{\eta}{2}[||\gamma - x + M(B\alpha+\mu)||^2 + ||\alpha-\beta||^2] \\ &s.t. \quad ||C_i(B\alpha+\mu)||_2^2 = L_i, i = 1,2,\ldots,m \end{aligned}$$

根据上文ADM,LADMAP算法描述,开始进行迭代求解(以下用ADM形式是因为γ,β可以得到闭合解)

4.1.1 γ与β的更新

舍弃拉格朗日增强方程中与γ,β无关的变量后

$$\gamma^{k+1} = \underset{\gamma}{argmin}\, ||\gamma||_1 + \frac{\eta_k}{2}||\gamma - [x - M(B\alpha^k+\mu) - \frac{\lambda_1^k}{\eta_k}]||^2$$

$$\beta^{k+1} = \underset{\beta}{argmin}\, ||\beta||_1 + \frac{\eta_k}{2\theta}||\beta - (\frac{\lambda_2^k}{\eta_k} + \alpha^k)||^2$$

是无约束优化问题,可以符合DCP规则采用CVXPY直接求解,但以下有更简单的方法

参考 [16](第4页),对于

$$prox_{f_i,\sigma}(w) = \underset{x_i}{argmin}\, f_i(x_i) + \frac{\sigma}{2}||x_i - w||^2$$

当f_i为1-范数时

$$prox_{f_i,\sigma}(w) = sgn(w)max(|w| - \sigma^{-1}) \tag{10}$$

$\gamma^{k+1}, \beta^{k+1}$带入后可以得到闭合解形式

4.1.2 α的求解

舍弃拉格朗日增强方程中与α无关的变量后

$$\begin{aligned} \alpha^{k+1} = \underset{\alpha}{argmin} \quad & z^T W z \\ s.t. \quad & z^T\Omega_i z = 0, i = 1,\ldots,m \end{aligned}$$

$$z = \begin{pmatrix} \alpha^T & 1 \end{pmatrix}^T$$

$$W = \begin{pmatrix} B^TM^TMB + I & 0 \\ 2[(\gamma^{k+1} - x + M\mu + \frac{\lambda_1^k}{n_k})^TMB - \beta^{k+1} + \frac{\lambda_2^k}{\eta_k} & 0 \end{pmatrix}$$

$$\Omega_i = \begin{pmatrix} B^TC_i^TC_iB & B^TC_i^TC_i\mu \\ \mu^TC_i^TC_iB & \mu^TC_i^TC_i\mu - L_i \end{pmatrix}$$

由 [17]中公式(3.89)

$$x^TAy = tr(x^TAy) = tr(Ayx^T)$$

可以改写形式

$$Q = zz^T$$

$$\begin{aligned} \min_{Q} \quad & tr(WQ) \\ s.t. \quad & tr(\Omega_iQ) = 0, i = 1,\ldots,m \\ & Q \geq 0, rank(Q) \leq 1 \end{aligned}$$

为借用ADMM方法,引入P

$$\begin{aligned} \min_{Q,P} \quad & tr(WQ) \\ s.t. \quad & tr(\Omega_iQ) = 0, i = 1,\ldots,m \\ & P = Q, rank(P) \leq 1, P \geq 0 \end{aligned}$$

其增强拉格朗日方程为

$$L_2(Q,P,G,\delta) = tr(WQ) + tr(G^T(Q-P)) + \frac{\delta}{2}||Q-P||_F^2$$

Q更新 Q是对称矩阵

$$\begin{aligned} Q^{l+1} = & \, argmin \;\; L_2(Q,P^l,G^l,\delta_l) \\ & tr(\Omega_iQ) = 0, i = 1,\ldots,m \end{aligned}$$

符合DCP规则,故直接用CVXPY求解决

P更新 限定P是对称矩阵

$$\begin{aligned} P^{l+1} = & \, argmin\, ||P - (Q^{l+1} + \frac{2}{\delta_l}G^l)|| \\ & P \geq 0 \\ & rank(P) \leq 1 \end{aligned}$$

设$\overline{Q} = Q^{l+1} + \frac{2}{\delta_l}G^l$,在P是对称矩阵条件下

$$\underset{P}{argmin}\, ||P - \overline{Q}||_F^2 = \underset{P}{argmin}\, ||P - \frac{\overline{Q}^T + \overline{Q}}{2}||_F^2$$

设$S = \frac{\overline{Q}^T+\overline{Q}}{2}$,P为半正定对称矩阵,且秩为1,则$P = \sigma vv^T, \sigma \geq 0$(4.3节,求解$\alpha$部分有解释),令$S$最大特征值为$\sigma_1$

$$\begin{aligned}||P-S||_F^2 =& ||P||_F^2 + ||S||_F^2 - 2tr(P^TS)\\ &\geq \sigma^2 + \sum_{i=1}^{n}\sigma_i^2 - 2\sigma\sigma_1\\ &= (\sigma-\sigma_1)^2 + \sum_{i=2}^{n}\sigma_i^2\\ &\geq \sum_{i=2}^{n}\sigma_i^2 + min(\sigma,0)^2\end{aligned}$$

当$\sigma = max(\sigma_1, 0), v = v_1$时等号成立,所以

$$P = max(\sigma_1, 0)v_1v_1^T$$

G更新(LAMAP)

$$G^{l+1} = G^l + \delta^l(Q^{l+1} - P^{l+1})$$

δ更新(LAMAP)

$$\delta^{l+1} = min(\delta^l\rho, \delta^{max})$$

其中$\rho \geq 1$且δ^{max}是常数

4.1.3 λ_1更新

$$\lambda_1^{k+1} = \lambda_1^k + \eta^k(\gamma^{k+1} - x + M(B\alpha^{k+1} + \mu))$$

4.1.4 λ_2更新

$$\lambda_2^{k+1} = \lambda_2^k + \eta^k(\alpha^{k+1} - \beta^{k+1})$$

4.1.5 η更新

$$\eta^{k+1} = min(\eta^k\rho, \eta^{max})$$

其中$\rho \geq 1$且η^{max}是常数

4.2 相机参数求解

同上,为采用LADMPSAP方法 [16],引入R,改写原问题为

$$\begin{aligned}\min_{R,m_1,m_2}\quad & ||R||_1\\ s.t.\quad & R = X - \begin{pmatrix}m_1^T\\ m_2^T\end{pmatrix}Y,\ m_1^Tm_2 = 0\end{aligned}$$

增广拉格朗日方程为

$$\begin{aligned}&L_3(R, m_1, m_2, H, \zeta, \tau)\\ &= ||R||_1 + tr\left(H^T\left[\begin{pmatrix}m_1^T\\ m_2^T\end{pmatrix}Y + R - X\right]\right) + \zeta(m_1^Tm_2)\\ &+ \frac{\tau}{2}\left[\left|\left|\begin{pmatrix}m_1^T\\ m_2^T\end{pmatrix}Y + R - X\right|\right|_F^2 + (m_1^Tm2)^2\right]\end{aligned}$$

R求解

$$R^{k+1} = \underset{R}{argmin}\,||R||_1 + \frac{\tau_k}{2}\left|\left|R + \begin{pmatrix}(m_1^k)^T\\ (m_2^k)^T\end{pmatrix}Y - X + \frac{H^k}{\tau_k}\right|\right|_F^2$$

同样,参考公式(10)求解

m_1求解

$$\begin{aligned}m_1^{k+1} =& \underset{m_1}{argmin}\left|\left|\begin{pmatrix}m_1^T\\ (m_2^k)^T\end{pmatrix}Y + R^{k+1} - X + \frac{H^k}{\tau_k}\right|\right|_F^2\\ &+ (m_1^Tm_2^k + \frac{\zeta^k}{\tau_k})^2\end{aligned}$$

符合DCP规则,可以用CVXPY求解,其次通过观察可知目标函数是个平滑的凸函数,故通过令一阶导数为零可以得到闭合解.m_2同理

5 编程具体实现中的细节

5.1 数据处理

5.1.1 3D姿态标准化

对于世界坐标系中的y,先将其变换到以物体为基准的坐标系,然后标准化

- 以左肩到右肩为x轴,
- 以两肩中点到腰为y轴
- z轴是x与y的叉乘.
- 然后以各个关节点各轴上点取平均得到$\mu \in R^3$,以其为原点.
- 对于每个关节点,求x_i, y_i, z_i坐标的平方和$s_i = x_i^2 + y_i^2 + z_i^2$,令$S = \sqrt{\sum_i s_i}$,$S$作为归一化系数,对于每个关节点,坐标变换为$x_i/S, y_i/S, z_i/S$

5.1.2 2D姿态标准化

同3D姿态标准化处理

- 然后以各个关节点各轴上点取平均得到$\mu \in R^2$,以其为原点.
- 对于每个关节点,求x_i, y_i坐标的平方和$s_i = x_i^2 + y_i^2$,令$S = \sqrt{\sum_i s_i}$,S作为归一化系数,对于每个关节点,坐标变换为$x_i/S, y_i/S$

5.2 字典学习

字典学习采用spams库提供算法,算出B,注意其中3D坐标已经按上述归一化,得到字典矩阵B为45×200

```
import spams
param = { 'K' : 200, 'lambda1' : 0.01, 'iter' :300}
B=spams.trainDL(np.asfortranarray(poses3Ds),**param)
```

5.3 由2D关节点信息推断3D姿态

获得骨骼长度 实现中,首先要确定各个关节点间距离,即骨骼长度.可人工设置,实验中直接由3D姿态坐标计算出.

获得初始α 字典学习中得到的字典矩阵B为45×200,$42(3\times15)$是关节点坐标,所以α是200×1的稀疏向量.迭代过程中需要初始化,如果以200维数据开始优化,效率低下.所以先用快速求解lasso问题的算法得出大概的α,再在此基础上开始优化求解.

$$\min_{\alpha} 0.5||x - MB\alpha||_2^2 + \lambda_1||\alpha||_1$$

```
a = spams.lasso(np.asfortranarray(x),np.asfortranarray(M@B),
     return_reg_path = False,lambda1=0.01)
```

得到α是稀疏的,.得到预估的α后,不再考虑为0的维度.如果此处引入了过大误差,后续算法精度难以得到保证.通过设置λ_1可以控制稀疏程度(5-30个较合适)

求解γ, β γ, β如文中所述利用10求解,

```
def soft(w, T):
    sigmaReverse=1/(2*T)
    return np.multiply(np.sign(w),np.fmax(abs(w)-sigmaReverse,0))
```

求解α 注意实践中Q, P收敛后,已知Q,根据推导,

$$Q = zz^T \begin{pmatrix} \alpha \\ 1 \end{pmatrix} \begin{pmatrix} \alpha^T & 1 \end{pmatrix} = \begin{pmatrix} \alpha\alpha^T & \alpha \\ \alpha^T & 1 \end{pmatrix}$$

α可以直接取自Q的最后一列,然而,实践中求出的Q往往没有满足$Q = zz^T$,从而造成了误差.可以通过分析特征值降低误差.

- 理想情况下

$$z \in R^{n\times1}$$

$$z = UDV^T, U \in R^{n\times n}, V^T = [1], D = \begin{pmatrix} \sigma_0 \\ \mathbf{0} \end{pmatrix}$$

$$z = \sigma_0 U(:0)$$

$$Q = zz^T = UD_Q U^T = UD^2U^T$$

$$D_Q = \begin{pmatrix} \sigma_0^2 & 0 \\ 0 & \mathbf{0} \end{pmatrix} = \begin{pmatrix} \sigma_{Q0} & 0 \\ 0 & \mathbf{0} \end{pmatrix}, \mathbf{0} \in R^{n-1\times n-1}$$

$$z = \sqrt{\sigma_{Q0}} U(:0)$$

- 实际情况中

$$D_Q = \begin{pmatrix} \sigma_{Q0} & 0 & \dots & 0 \\ 0 & \sigma_{Q1} & \dots & 0 \\ \vdots & \ddots & \vdots & \\ 0 & 0 & \dots & \sigma_{Q(n-1)} \end{pmatrix}$$

$$i = \underset{i}{argmax} \quad \sigma_{Qi}$$

估计z的值为

$$z = \sqrt{\sigma_{Qi}} U(:i)$$

5.4 推断出3D姿态后求解相机参数

m_1, m_2是凸函数,可以通过导数为零的方式求解

```
def update_m1(m1, m2, R, X, H, Y, xi, tao):
    x,y,z=symbols('x y z', real=True)
    m10 = np.mat([x, y, z]).T
    F0 = (Matrix(np.vstack((np.transpose(m10), np.transpose(m2)))
         + R - X + H/tao).norm rd='fro'))**2
+ Matrix((m10.T @ m2 + xi/tao)**2).norm()
    m1=[]
    m10=solve([diff(F0,x),diff(F0,y),diff(F0,z)],[x,y,z])
    m1.append(float(m10[x]))
    m1.append(float(m10[y]))
    m1.append(float(m10[z]))
    return np.mat(m1).T
```

R同样可以使用10方法求解

5.5 相机参数求解方法改进

实践中发现利用迭代优化方法求解相机参数效率低下,且精度不高,以下采用一种有闭合解的近似方法.加速了算法,且提升了精度.

弱透视投影

$$P=\begin{pmatrix}s_1&0&0&0\\0&s_2&0&0\\0&0&0&1\end{pmatrix}\begin{pmatrix}\boldsymbol{R'}&\boldsymbol{t}\\\boldsymbol{0}^T&1\end{pmatrix}$$
$$=\begin{pmatrix}s_1&0&0\\0&s_2&0\\0&0&1\end{pmatrix}\begin{pmatrix}\boldsymbol{r}_1^T&t_1\\\boldsymbol{r}_2^T&t_2\\\boldsymbol{0}^T&1\end{pmatrix}$$

$$\begin{pmatrix}x\\y\end{pmatrix}=P\begin{pmatrix}X\\Y\\Z\\1\end{pmatrix}=\begin{pmatrix}s_1\boldsymbol{r}_1^T\\s_1\boldsymbol{r}_2^T\end{pmatrix}\begin{pmatrix}X\\Y\\Z\end{pmatrix}+\begin{pmatrix}s_1t_1\\s_2t_2\end{pmatrix}$$
$$=\begin{pmatrix}s_1&0\\0&s_2\end{pmatrix}\begin{pmatrix}\boldsymbol{r_1}^T\\\boldsymbol{r_2}^T\end{pmatrix}\begin{pmatrix}X\\Y\\Z\end{pmatrix}+\begin{pmatrix}t_1'\\t_2'\end{pmatrix}$$

因为3D坐标和2D坐标都是以均值点作为原点,所以t_1,t_2为0

$$\begin{pmatrix}x\\y\end{pmatrix}=\boldsymbol{s}\begin{pmatrix}\boldsymbol{r_1}^T\\\boldsymbol{r_2}^T\end{pmatrix}\begin{pmatrix}X\\Y\\Z\end{pmatrix}=\boldsymbol{sR}\begin{pmatrix}X\\Y\\Z\end{pmatrix}=\boldsymbol{M}\begin{pmatrix}X\\Y\\Z\end{pmatrix}$$

近似求弱透视矩阵$\boldsymbol{M}$ 假设$x\in R^2$,$y\in R^3$,理想情况下

$$\boldsymbol{M}y=x$$

当$Ax=b$,无解,即A逆不存在时,最佳近似解为$x=A^{\dagger}b$,所以近似估计$\boldsymbol{M}$为

$$\boldsymbol{M'}=xy^T(yy^T)^{-1}$$

一般条件下,这样得出的矩阵不满足M是正交矩阵的条件,且$\boldsymbol{M'}y\neq x$

- **方法1** 对$\boldsymbol{M'}$求RQ分解

$$\boldsymbol{M'}=\begin{pmatrix}0&s_1'&\eta\\0&0&s_2'\end{pmatrix}\begin{pmatrix}\boldsymbol{r_3'}^T\\\boldsymbol{r_1'}^T\\\boldsymbol{r_2'}^T\end{pmatrix}$$

实验中η相对s_1,s_2较小,且接近于0,所以令$\eta=0$,再反向求M得到估计值

$$\boldsymbol{M'}=\begin{pmatrix}0&s_1'&0\\0&0&s_2'\end{pmatrix}\begin{pmatrix}\boldsymbol{r_3'}^T\\\boldsymbol{r_1'}^T\\\boldsymbol{r_2'}^T\end{pmatrix}$$
$$=\begin{pmatrix}s_1'&0\\0&s_2'\end{pmatrix}\begin{pmatrix}\boldsymbol{r_1'}^T\\\boldsymbol{r_2'}^T\end{pmatrix}$$

代码

```
def CameraEstimation(X,Y):
    M=X@Y.T@np.linalg.inv(Y@Y.T)
    r,q=linalg.rq(M)
    r[0,2]=0.05*r[0,2]
    M=r@q
    return M
```

- **方法2** 对理想的$\boldsymbol{M}$,有

$$\boldsymbol{M}=\begin{pmatrix}\boldsymbol{m_1}\\\boldsymbol{m_2}\end{pmatrix},\ 且\boldsymbol{m_1}^T\boldsymbol{m_2}=0$$

$$M=\boldsymbol{U\Sigma V}^T=\boldsymbol{U}\begin{pmatrix}s_1&0&0\\0&s_2&0\end{pmatrix}\boldsymbol{V}^T$$

$$\boldsymbol{MM}^T=\begin{pmatrix}\boldsymbol{m_1}\\\boldsymbol{m_2}\end{pmatrix}\begin{pmatrix}\boldsymbol{m_1}^T&\boldsymbol{m_2}^T\end{pmatrix}=\begin{pmatrix}\boldsymbol{m_1m_1}^T&\boldsymbol{m_1m_2}^T\\\boldsymbol{m_2m_1}^T&\boldsymbol{m_2m_2}^T\end{pmatrix}$$
$$=\begin{pmatrix}\sigma_1&0\\0&\sigma_2\end{pmatrix}$$

$$\boldsymbol{MM}^T=\boldsymbol{U\Sigma}^2\boldsymbol{U}^T=\boldsymbol{U}\begin{pmatrix}s_1^2&0\\0&s_2^2\end{pmatrix}\boldsymbol{U}^T$$

所以有$\boldsymbol{U}=\boldsymbol{I}$,故

$$\boldsymbol{M}=\boldsymbol{U}\begin{pmatrix}s_1&0&0\\0&s_2&0\end{pmatrix}\boldsymbol{V}^T=\begin{pmatrix}s_1&0\\0&s_2\end{pmatrix}\boldsymbol{U}\begin{pmatrix}1&0&0\\0&1&0\end{pmatrix}\boldsymbol{V}^T=\boldsymbol{sR}$$

对于实际的$\boldsymbol{M'}$,先进行SVD分解$\boldsymbol{U'}$,$\begin{pmatrix}s_1'&0&0\\0&s_2'&0\end{pmatrix}$,$\boldsymbol{V}^T$

$$\boldsymbol{M}=\boldsymbol{s'R'}$$

其中$\boldsymbol{s'}=\begin{pmatrix}s_1'&0\\0&s_2'\end{pmatrix}$,$\boldsymbol{R'}=\boldsymbol{V}^T$或$\boldsymbol{R'}=\boldsymbol{U'}\begin{pmatrix}1&0&0\\0&1&0\end{pmatrix}\boldsymbol{V'}^T$,后者效果理论与实践上有较好结果

6 总结

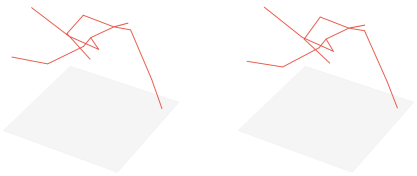
优点:

- 相对于深度学习,其学习速度快.可以用于小样本的情况
- 学习过程中,不需要2D图像和3D姿态信息对,只需要3D姿态信息,数据相对容易获取.

缺点:

- 许多参数需要细致设置.如果收敛条件没有设置好,实践中出现了发散的情况.
- 推断速度慢,迭代求解,可并行化程度低.

- 需要人工设置的各个关节点间距离,即骨骼长度.
  工程上完整实现了该算法涉及的算法过程,实现部分trick参考了原作者通过邮箱提供的非正式版本Matlab源码.实现精度结果达到和原文几乎一致.



某次测试中,左边是ground truth,右边由算法重建出的效果

参考文献

[1] Wei S E, Ramakrishna V, Kanade T, et al. Convolutional pose machines. In: Proc of Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. 2016, 4724–4732

[2] Newell A, Yang K, Deng J. Stacked hourglass networks for human pose estimation. In: Proc of European conference on computer vision. Springer, 2016, 483–499

[3] Martinez J, Hossain R, Romero J, et al. A Simple Yet Effective Baseline for 3d Human Pose Estimation. In: Proc of 2017 IEEE International Conference on Computer Vision (ICCV). Venice: IEEE, 2017, 2659–2668

[4] Fang H S, Xu Y, Wang W, et al. Learning Pose Grammar to Encode Human Body Configuration for 3D Pose Estimation. 8

[5] Zhao L, Peng X, Tian Y, et al. Semantic Graph Convolutional Networks for 3D Human Pose Regression. In: Proc of 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Long Beach, CA, USA: IEEE, 2019, 3420–3430

[6] Ci H, Wang C, Ma X, et al. Optimizing Network Structure for 3D Human Pose Estimation. In: Proc of 2019 IEEE/CVF International Conference on Computer Vision (ICCV). Seoul, Korea (South): IEEE, 2019, 2262–2271

[7] Zhou X, Huang Q, Sun X, et al. Towards 3D Human Pose Estimation in the Wild: a Weakly-supervised Approach. arXiv:1704.02447 [cs], 2017. ArXiv: 1704.02447

[8] Yang W, Ouyang W, Wang X, et al. 3D Human Pose Estimation in the Wild by Adversarial Learning. arXiv:1803.09722 [cs], 2018. ArXiv: 1803.09722

[9] Chen C H, Tyagi A, Agrawal A, et al. Unsupervised 3D Pose Estimation With Geometric Self-Supervision. In: Proc of 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Long Beach, CA, USA: IEEE, 2019, 5707–5717

[10] Rhodin H, Salzmann M, Fua P. Unsupervised Geometry-Aware Representation for 3D Human Pose Estimation. In: Ferrari V, Hebert M, Sminchisescu C, et al, (eds.). Proc of Computer Vision – ECCV 2018. Cham: Springer International Publishing, 2018: 765–782

[11] Chen X, Lin K Y, Liu W, et al. Weakly-Supervised Discovery of Geometry-Aware Representation for 3D Human Pose Estimation. arXiv:1903.08839 [cs], 2019. ArXiv: 1903.08839

[12] Safonova A, Hodgins J K, Pollard N S. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. ACM Transactions on Graphics (ToG), 2004, 23(3):514–521. Publisher: ACM New York, NY, USA

[13] Mairal J, Bach F, Ponce J, et al. Online dictionary learning for sparse coding. In: Proc of Proceedings of the 26th annual international conference on machine learning. 2009, 689–696

[14] Boyd S. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. Foundations and Trends® in Machine Learning, 2010, 3(1):1–122

[15] Lin Z, Liu R, Su Z. Linearized alternating direction method with adaptive penalty for low-rank representation. In: Proc of Advances in neural information processing systems. 2011, 612–620

[16] Lin Z, Liu R, Li H. Linearized alternating direction method with parallel splitting and adaptive penalty for separable convex programs in machine learning. Machine Learning, 2015, 99(2):287–325

[17] Gentle J E. Matrix Algebra. Springer Texts in Statistics. Cham: Springer International Publishing, 2017