projection_factor

一 单位球模式

残差计算

- 先求与P_l^{cj}垂直的一组正交基底，将之记录为B
- 将P_l^{ci}转换到P^{cj}这个frame下面

$$P_l^{cj'} = R_c^{b^T}(R_{bj}^T(R_{bi}(R_c^b P_l^{ci} + t_c^b) + t_{bi} - t_{bj}) - t_c^b)$$

- 计算$P_l^{cj'}$和P_l^{cj}在单位球上的距离，然后将这个距离投影到B这个基底上

$$r' = B(\frac{P_l^{cj'}}{||P_l^{cj'}||} - \frac{P_l^{cj}}{||P_l^{cj}||})$$

- 将得到的残差r'乘上协方差，注意这里的协方差是固定的，也就是相当于一个固定的权

$$r = sqrt\_info * r'$$

雅克比矩阵的计算

在vins的代码中，将这里待优化的量分成了四个块，分别为

- $X_0 = [t_{bi}, q_{bi}]^T$，3+4=7
- $X_1 = [t_{bj}, q_{bj}]^T$，3+4=7
- $X_2 = [t_c^i, q_c^i]^T$，3+4=7
- $X_3 = \lambda = \frac{1}{d}$,指的是在$ci$这个frame下的深度，1

理论上来说，这里求的雅克比矩阵应该有四个。

每一个雅克比应该都是$J_i = \frac{\partial r}{\partial X_i}$，可以使得

$$f(X_i \boxplus \delta X_i) = f(X_i) + J_i \delta X_i$$

但是，实际上由于这里的q是四元数，是一种overparam的表示，我们给它定义了一种更新的方式，所以最后应该是

$$f(X_i \boxplus \Delta_i) = f(X_i) + J_i D_i \Delta_i$$

即，这里的Δ的维度比X_i的维度要小，例如这里是3，而原本维度是4维的。

而$D_i = \frac{\delta X_i}{\partial \Delta_i}$,也就是说

$$\begin{aligned} f(X_i \boxplus \Delta) &= f(X_i) + \frac{\partial r}{\partial X_i}\frac{\delta X_i}{\partial \Delta}\Delta = f(X_i) + J_i D_i \Delta_i && \text{(a)} \\ &= f(X_i) + \frac{\partial r}{\partial \Delta}\Delta = f(X_i) + F_i \Delta_i && \text{(b)} \end{aligned}$$

理论上我们可以直接求出式子(b)所用到的雅克比F_i的形式，但是由于ceres设计上的原因，使得我们必须使用(a)的形式，即分成ProjectionFactor里面的Evaluate里的雅克比J_i和PoseLocalParameterization里面的ComputeJacobian里的D_i的形式。这样子效率会比较低，而且可能形式会比较麻烦？反正我们可以一步求出F_i的形式，所以，代码里面用了一些技巧

即令$J_i = [F_i\ 0]\ D_i = [I\ 0]^T$,此时，$J_i * D_i = F_i + 0 = F_i$,依然等价，这也是代码里面做的骚操作，所以一开始看有点懵逼。

> 参考链接:
>
> 博客园
>
> 神秘资料

综上，所以我们实际上要求的雅克比是$F_i = \frac{\partial r}{\partial \Delta}$的形式。

即待优化的变量我们，要做出一点改变

- $X_0 = [t_{bi}, q_{bi}]^T$，3+4=7 ==> $X_0 = [t_{bi}, \theta_{bi}]^T$,3+3=6
- $X_1 = [t_{bj}, q_{bj}]^T$，3+4=7 ==> $X_1 = [t_{bj}, \theta_{bj}]^T$,3+3=6
- $X_2 = [t_c^b, q_c^b]^T$，3+4=7 ==> $X_2 = [t_c^b, \theta_c^b]^T$,3+3=6
- $X_3 = \lambda = \frac{1}{d}$,指的是在$ci$这个frame下的深度，1,不变

$$
\begin{aligned}
F_i =& \frac{\partial r}{\partial \Delta} sqrt\_info * B * \frac{\partial r'}{\partial \Delta} \\
=& sqrt\_info * B \frac{\partial (\frac{P_l^{cj'}}{||P_l^{cj'}||} - \frac{P_l^{cj}}{||P_l^{cj}||})}{\partial \Delta} \\
=& sqrt\_info * B \frac{\partial \frac{P_l^{cj'}}{||P_l^{cj'}||}}{\partial \Delta}
\end{aligned}
$$

$sqrt_info$和B都是常数，直接乘上就好了,主要求最后部分$\frac{\partial \frac{P_l^{cj'}}{||P_l^{cj'}||}}{\partial \Delta}$

> 对于形如$\frac{d\frac{f(x)}{||f(x)||_2}}{dx}$的导数，用链式法则展开，可以得到:
>
> $$
> \begin{aligned}
> \frac{d\frac{f(x)}{||f(x)||_2}}{dx} &= \frac{df(x)}{dx}\frac{1}{||f(x)||_2} + f(x)\frac{d\frac{1}{||f(x)||_2}}{dx} \\
> &= \frac{df(x)}{dx}\frac{1}{||f(x)||_2} - f(x)\frac{1}{||f(x)||_2^2}\frac{d||f(x)||_2}{dx}
> \end{aligned}
> $$
>
> 再来看一下$\frac{d||f(x)||_2}{dx}$
>
> $$
> \begin{aligned}
> \frac{d||f(x)||_2}{dx} &= \frac{d[f(x)^T f(x)]^{\frac{1}{2}}}{dx} \\
> &= \frac{1}{2}[f(x)^T f(x)]^{-\frac{1}{2}}\frac{d[f(x)^T f(x)]}{dx}
> \end{aligned}
> $$
>
> 再看一下$\frac{d[f(x)^T f(x)]}{dx}$
>
> $$
> \frac{d[f(x)^T f(x)]}{dx} = 2f(x)^T\frac{df(x)}{dx}
> $$
>
> 综上，我们可以得到这样的形式
>
> $$
> \begin{aligned}
> \frac{d\frac{f(x)}{||f(x)||_2}}{dx} &= \frac{1}{||f(x)||_2}f'(x) - \frac{1}{||f(x)||_2^3}f(x)f(x)^T f'(x) \\
> &= (\frac{1}{||f(x)||_2}I - \frac{1}{||f(x)||_2^3}f(x)f(x)^T)f'(x)
> \end{aligned} \tag{6}
> $$
>
> 这段也是抄的博客园的推导，这波推导着实狠。

这里的$f(x)$为$P_l^{cj'}$，也是说，我们的导数形式变成了

$$
\begin{aligned}
F_i &= \frac{\partial r}{\partial \Delta} sqrt\_info * B * \frac{\partial r'}{\partial \Delta} \\
&= sqrt\_info * B \frac{\partial(\frac{P_l^{cj'}}{||P_l^{cj'}||} - \frac{P_l^{cj}}{||P_l^{cj}||})}{\partial \Delta} \\
&= sqrt\_info * B \frac{\partial \frac{P_l^{cj'}}{||P_l^{cj'}||}}{\partial \Delta} \\
&= sqrt\_info * B * (\frac{1}{||P_l^{cj'}||_2} I - \frac{1}{||P_l^{cj'}||_2^3} P_l^{cj'} P_l^{cj'T})(P_l^{cj'})'
\end{aligned}
$$

```
double norm = pts_camera_j.norm();
Eigen::Matrix3d norm_jaco;
double x1, x2, x3;
x1 = pts_camera_j(0);
x2 = pts_camera_j(1);
x3 = pts_camera_j(2);
norm_jaco << 1.0 / norm - x1 * x1 / pow(norm, 3), -x1 * x2 / pow(norm, 3), -x1 *
x3 / pow(norm, 3),
-x1 * x2 / pow(norm, 3), 1.0 / norm - x2 * x2 / pow(norm, 3), -x2 * x3 /
pow(norm, 3),
-x1 * x3 / pow(norm, 3), -x2 * x3 / pow(norm, 3), 1.0 / norm - x3 * x3 /
pow(norm, 3);
reduce = tangent_base * norm_jaco;

reduce = sqrt_info * reduce;
```

- `pts_camera_j`为$P_l^{cj'}$
- `norm`为$||P_l^{cj'}||_2$
- `tangent_base`为B

剩下的就是求$(P_l^{cj'})'$

对第一个参数块求导

$$
X_0 = [t_{bi}, \theta_{bi}]^T
$$

$$
\begin{aligned}
(P_l^{cj'})' &= \frac{dP_l^{cj'}}{dX_0} \\
&= [\frac{dP_l^{cj'}}{dt_i}, \frac{dP_l^{cj'}}{d\theta_{bi}}]
\end{aligned}
$$

$$
\frac{dP_l^{cj'}}{dt_{bi}} = \frac{d\left(R_c^{b\,T}(R_{bj}^T(R_{bi}(R_c^b P_l^{ci} + t_c^b) + t_{bi} - t_{bj}) - t_c^b)\right)}{dt_{bi}} = R_c^{b\,T} R_{bj}^T
$$

```
jaco_i.leftCols<3>() = ric.transpose() * Rj.transpose();
```

$$
\begin{aligned}
\frac{dP_l^{cj'}}{d\theta_{bi}} &= \frac{d\left(R_c^{b^T}(R_{bj}^T(R_{bi}(R_c^b P_l^{ci}+t_c^b)+t_{bi}-t_{bj})-t_c^b)\right)}{d\theta_{bi}} \\
&= \frac{d\left(R_c^{b^T}R_{bj}^T R_{bi}(R_c^b P_l^{ci}+t_c^b)\right)}{d\theta_{bi}} \\
&= \frac{R_c^{b^T}R_{bj}^T R_{bi}Exp(d\theta_{bi})(R_c^b P_l^{ci}+t_c^b)-R_c^{b^T}R_{bj}^T R_{bi}(R_c^b P_l^{ci}+t_c^b)}{d\theta_{bi}} \\
&= \frac{R_c^{b^T}R_{bj}^T R_{bi}\left(I+\lfloor d\theta_{bi}\rfloor_\times\right)(R_c^b P_l^{ci}+t_c^b)-R_c^{b^T}R_{bj}^T R_{bi}(R_c^b P_l^{ci}+t_c^b)}{d\theta_{bi}} \\
&= \frac{R_c^{b^T}R_{bj}^T R_{bi}\left(\lfloor d\theta_{bi}\rfloor_\times\right)(R_c^b P_l^{ci}+t_c^b)}{d\theta_{bi}} \\
&= \frac{R_c^{b^T}R_{bj}^T R_{bi}\left(-\lfloor(R_c^b P_l^{ci}+t_c^b)\rfloor_\times\right)d\theta_{bi}}{d\theta_{bi}} \\
&= R_c^{b^T}R_{bj}^T R_{bi}\left(-\lfloor(R_c^b P_l^{ci}+t_c^b)\rfloor_\times\right)
\end{aligned}
$$

```
jaco_i.rightCols<3>() = ric.transpose() * Rj.transpose() * Ri * -Utility::skewSymmetric(pts_imu_i);
```

对第二个参数块求导

$$X_1 = [t_{bj}, \theta_{bj}]^T$$

$$
\begin{aligned}
(P_l^{cj'})' &= \frac{dP_l^{cj'}}{dX_1} \\
&= [\frac{dP_l^{cj'}}{dt_j}, \frac{dP_l^{cj'}}{d\theta_{bj}}]
\end{aligned}
$$

$$\frac{dP_l^{cj'}}{dt_{bj}} = \frac{d\left(R_c^{b^T}(R_{bj}^T(R_{bi}(R_c^b P_l^{ci}+t_c^b)+t_{bi}-t_{bj})-t_c^b)\right)}{dt_{bi}} = -R_c^{b^T}R_{bj}^T$$

```
jaco_j.leftCols<3>() = ric.transpose() * -Rj.transpose();
```

$$
\begin{aligned}
\frac{dP_l^{cj'}}{d\theta_{bj}} &= \frac{d\left(R_c^{b^T}(R_{bj}^T(R_{bi}(R_c^b P_l^{ci}+t_c^b)+t_{bi}-t_{bj})-t_c^b)\right)}{d\theta_{bj}} \\
&= \frac{d\left(R_c^{b^T}(R_{bj}^T(R_{bi}(R_c^b P_l^{ci}+t_c^b)+t_{bi}-t_{bj}))\right)}{d\theta_{bj}} \\
&= \frac{d\left(R_c^{b^T}((-\lfloor d\theta_j\rfloor_\times)(R_{bj}^T(R_{bi}(R_c^b P_l^{ci}+t_c^b)+t_{bi}-t_{bj}))\right)}{d\theta_{bj}} \\
&= \frac{d\left(R_c^{b^T}(\lfloor(R_{bj}^T(R_{bi}(R_c^b P_l^{ci}+t_c^b)+t_{bi}-t_{bj})\rfloor_\times)d\theta_j\right)}{d\theta_{bj}} \\
&= R_c^{b^T}(\lfloor(R_{bj}^T(R_{bi}(R_c^b P_l^{ci}+t_c^b)+t_{bi}-t_{bj})\rfloor_\times)
\end{aligned}
$$

又乘括号里面的东西其实就是将P_l^{ci}转化成P_l^{bj}

```
jaco_j.rightCols<3>() = ric.transpose() * Utility::skewSymmetric(pts_imu_j);
```

对第三个参数块求导

$$X_2 = [t_c^b, \theta_c^b]^T$$

$$\begin{aligned}({P_l^{cj}}')' =& \frac{d{P_l^{cj}}'}{dX_1} \\ =& [\frac{d{P_l^{cj}}'}{dt_c^b}, \frac{d{P_l^{cj}}'}{d\theta_c^b}]\end{aligned}$$

$$\frac{d{P_l^{cj}}'}{dt_c^b} = \frac{d\left({R_c^b}^T(R_{bj}^T(R_{bi}(R_c^b P_l^{ci} + t_c^b) + t_{bi} - t_{bj}) - t_c^b)\right)}{dt_c^b} = {R_c^b}^T R_{bj}^T R_c^b - {R_c^b}^T$$

```
jaco_ex.leftCols<3>() = ric.transpose() * (Rj.transpose() * Ri -
Eigen::Matrix3d::Identity());
```

$$\begin{aligned}\frac{d{P_l^{cj}}'}{d\theta_c^b} =& \frac{d\left({R_c^b}^T(R_{bj}^T(R_{bi}(R_c^b P_l^{ci} + t_c^b) + t_{bi} - t_{bj}) - t_c^b)\right)}{d\theta_c^b} \\ =& \frac{d\left({R_c^b}^T R_{bj}^T R_{bi} R_c^b P_l^{ci} + {R_c^b}^T R_{bj}^T R_{bi} t_c^b + {R_c^b}^T R_{bj}^T t_{bi} - {R_c^b}^T R_{bj}^T t_{bj} - {R_c^b}^T t_c^b\right)}{d\theta_c^b} \\ =& \frac{d\left({R_c^b}^T R_{bj}^T R_{bi} R_c^b P_l^{ci}\right)}{d\theta_c^b} + \frac{d\left({R_t^b}^T(R_{bj}^T(t_{bi} - t_{bj} + R_{bi} t_c^b) - t_c^b)\right)}{d\theta_c^b} \\ =& \dot{R_c^b}^T R_{bj}^T R_{bi} R_c^b P_l^{ci} + {R_c^b}^T R_{bj}^T R_{bi} \dot{R}_c^b P_l^{ci} + \lfloor {R_t^b}^T(R_{bj}^T(t_{bi} - t_{bj} + R_{bi} t_c^b) - t_c^b) \rfloor_\times \\ =& \lfloor {R_c^b}^T {R_c^b}^T R_{bj}^T R_{bi} R_c^b P_l^{ci} \rfloor_\times - {R_c^b}^T R_{bj}^T R_{bi} R_c^b \lfloor P_l^{ci} \rfloor_\times + \lfloor {R_t^b}^T(R_{bj}^T(t_{bi} - t_{bj} + R_{bi} t_c^b) - t_c^b) \rfloor_\times\end{aligned}$$

```
Eigen::Matrix3d tmp_r = ric.transpose() * Rj.transpose() * Ri * ric;
            jaco_ex.rightCols<3>() = -tmp_r *
Utility::skewSymmetric(pts_camera_i) + Utility::skewSymmetric(tmp_r *
pts_camera_i) +
                                    Utility::skewSymmetric(ric.transpose() *
(Rj.transpose() * (Ri * tic + Pi - Pj) - tic));
```

> 这个推导看着吓人，但是有耐心一步步拆开还是可以推导的

对第四个参数块求导

$$X_3 = \lambda$$

$$\begin{aligned}\frac{d{P_l^{cj}}'}{d\lambda} =& \frac{d\left({R_c^b}^T(R_{bj}^T(R_{bi}(R_c^b P_l^{ci} + t_c^b) + t_{bi} - t_{bj}) - t_c^b)\right)}{d\lambda} \\ =& \frac{d\left({R_c^b}^T R_{bj}^T R_{bi} R_c^b P_l^{ci}\right)}{d\lambda} \\ =& \frac{d\left({R_c^b}^T R_{bj}^T R_{bi} R_c^b \overline{P_l^{ci}} \frac{1}{\lambda}\right)}{d\lambda} \\ =& -{R_c^b}^T R_{bj}^T R_{bi} R_c^b \overline{P_l^{ci}} \lambda^{-2}\end{aligned}$$

```
jacobian_feature = reduce * ric.transpose() * Rj.transpose() * Ri * ric * pts_i
* -1.0 / (inv_dep_i * inv_dep_i);
```

二 非单位球

非单位球，主要是$r^{'}$发生了改变

$$r^{'}=\left[\frac{P_l^{cj^{'}}}{P_l^{cj^{'}}(2)}-\frac{P_l^{cj}}{P_l^{cj}(2)}\right]_{uv}$$

即取归一化平面上的前两维的差

$$\begin{aligned}
\frac{dr^{'}}{dX_i}&=\frac{d\left[\frac{P_l^{cj^{'}}}{P_l^{cj^{'}}(2)}\right]_{uv}}{dX_i}\\
&=\begin{bmatrix}1&0&0\\0&1&0\end{bmatrix}\frac{d\frac{P_l^{cj^{'}}}{P_l^{cj^{'}}(2)}}{dX_i}\\
&=\begin{bmatrix}1&0&0\\0&1&0\end{bmatrix}\frac{d\frac{f(x)}{f(x)(2)}}{dx}\\
&=\begin{bmatrix}1&0&0\\0&1&0\end{bmatrix}\frac{f^{'}(x)f(x)(2)-f^{'}(x)(2)f(x)}{f^2(x)(2)}\\
&=\begin{bmatrix}1&0&0\\0&1&0\end{bmatrix}\left(\frac{If(x)(2)-f(x)\begin{bmatrix}0&0&1\end{bmatrix}}{f^2(x)(2)}\right)f^{'}(x)\\
&=\begin{bmatrix}1&0&0\\0&1&0\end{bmatrix}\left(\begin{bmatrix}\frac{1}{z}&0&0\\0&\frac{1}{z}&0\\0&0&\frac{1}{z}\end{bmatrix}-\begin{bmatrix}0&0&\frac{x}{z^2}\\0&0&\frac{y}{z^2}\\0&0&\frac{z}{z^2}\end{bmatrix}\right)f^{'}(x)\\
&=\begin{bmatrix}\frac{1}{z}&0&-\frac{x}{z^2}\\0&\frac{1}{z}&-\frac{y}{z^2}\end{bmatrix}f^{'}(x)
\end{aligned}$$

这里的

$x=f^2(x)(0)$

$y=f^2(x)(1)$

$z=f^2(x)(2)$

于是，你就能够得到和代码一样的东西

```
reduce << 1. / dep_j, 0, -pts_camera_j(0) / (dep_j * dep_j),
            0, 1. / dep_j, -pts_camera_j(1) / (dep_j * dep_j);
reduce = sqrt_info * reduce;
```

三 总结

- 单位球和针孔模型求误差，都是一种策略，论文说的是用单位球可以兼容更多的相机模型。但是代码里面默认用的是针孔模型，如果想改的话可以将在parameters.h里面的`//#define UNIT_SPHERE_ERROR`解除注释即可。
- 两者的区别仅仅在于reduce这个矩阵，详细可以看上面的推导。

projection_td_factor

td版本主要就就是添加了对时间的偏移，而偏移的这段时间用的是对应时刻的速度去补充的。

待优化的量多了一个td。

所带来的区别就是

$$\overline{P}_{td} = \overline{P} - (td - td_{old} + \frac{r}{Row}TR) * v$$

也就是说，这里的td估计的是，相机的数据比IMU的数据慢了多少的意思。我们需要把这一部分数据给减回去。

td_{old}是指，在记录这个点的时候已知的td,新的td减去旧的td得到的是在这期间延迟增加了多少。也就是说，如果td一直保持不变，那么我们认为时间戳是对齐了的。

r是这个像素到多少行，Row是总行数，TR是卷帘快门成像所有行需要的时间，v是速度，注意，是像素/s

在这个过程中，其实发生变化的就是上面在没有td的版本中P_l^{ci}发生了变化而已，而且$P_l^{ci} = g(td)$是一个只和td有关的函数，也就是说，这个过程就会变得和没有td的版本很类似了.

残差

我们只需要用这个新的P_l^{ci}和P_l^{cj}代替进去就好了。

雅克比

第一个参数块

同上，只需要用这个新的P_l^{ci}代替进去就好了。

第二个参数块

同上，只需要用这个新的P_l^{ci}代替进去就好了。

第三个参数块

同上，只需要用这个新的P_l^{ci}代替进去就好了。

第四个参数块

同上，只需要用这个新的P_l^{ci}代替进去就好了。

第五个参数块

这个参数块是$X_4 = td$

采用链式法则

$$\begin{aligned}
\frac{dP_l^{cj'}}{dtd} &= \frac{dP_l^{cj'}}{dP_l^{ci}} \frac{dP_l^{ci}}{dtd} \\
&= \frac{d\left(R_c^{bT}(R_{bj}^T(R_{bi}(R_c^b P_l^{ci} + t_c^b) + t_{bi} - t_{bj}) - t_c^b)\right)}{dP_l^{ci}} \frac{d\frac{\overline{P} - (td - td_{old} + \frac{r}{Row}TR) * v_i}{\lambda}}{dtd} \\
&= R_c^{bT} R_{bj}^T R_{bi} R_c^b * \frac{(-v_i)}{\lambda}
\end{aligned}$$

$$\frac{d\frac{P_l^{cj}}{||P_l^{cj}||}}{dtd} = \frac{d\frac{P_l^{cj}}{1/\lambda}}{dtd} = -v_j$$

$$
\begin{aligned}
\frac{dr}{dX_4} &= sqrt\_info * B \frac{\partial(\frac{P_l^{cj'}}{||P_l^{cj'}||} - \frac{P_l^{cj}}{||P_l^{cj}||})}{\partial\Delta} \\
&= reduce * \frac{dP_l^{cj'}}{dtd} - sqrt\_info * B * \frac{d\frac{P_l^{cj}}{||P_l^{cj}||}}{dtd} \\
&= reduce * R_c^{b\,T} R_{bj}^T R_{bi} R_c^b * \frac{(-v_i)}{\lambda} + sqrt\_info * B * v_j \\
&= reduce * R_c^{b\,T} R_{bj}^T R_{bi} R_c^b * \frac{(-v_i)}{\lambda} + sqrt\_info * {v_j}_{xy}
\end{aligned}
$$

```
jacobian_td = reduce * ric.transpose() * Rj.transpose() * Ri * ric * velocity_i
/ inv_dep_i * -1.0  + sqrt_info * velocity_j.head(2);
```

- 关于最后一步的导数，必须要满足$||P_l^{cj}|| = 1/\lambda$才行。

> 由于这个是一个单位球模型，所以我们可以假设成像面和永远都和成像方向垂直，也就是说，此时$||P_l^{cj}|| = 1/\lambda$,虽然已经在v_j的方向上发生了一点点小位移，但是忽略不计。问题不大

- **||P^{cj}_l||**必须当成一个常量来处理。虽然不是严格意义上的常量

> 理论上来说，确实可以当成一个常量，毕竟一直是在v_j方向发生移动而已，类似旋转的效果

- 由于v_j也是和P_l^{cj}垂直，所以也就是在B这个平面上，然后假设速度v_j刚刚好和B是同一组基底。

> 这一点其实就比较疑惑了，除非v_j刚刚好是在B这个基底下进行定义的。虽然我觉得不可能，除非B恒定是$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$,也只能够假设是了，不然根本的不出来这个东西。

综上：我总觉得这个td模型在球形模型下哪里怪怪的

- 如果是在针孔模型下面，就没有这么多事情了

$$
\frac{P_l^{cj}}{P_l^{cj}(2)}_{uv} = \overline{P} - (td - td_{old} + \frac{r}{Row}TR) * v_j
$$

  直接求导，结果就一模一样了

$$
[\frac{dP_l^{cj}}{dP_l^{cj}(2)}_{uv}]_{xy} = -{v_j}_{xy}
$$