

Solution to the 6th Homework

Yimeng Zhu

December 15, 2019

Contents

1 LK 光流 (5 分, 约 3 小时)	2
1.1 光流文献综述 (1 分)	2
1.2 forward-addtive Gauss-Newton 光流的实现 (1 分)	2
1.3 反向法 (1 分)	3
1.4 推广至金字塔 (2 分)	3
1.5 讨论	3
2 直接法 (5 分, 约 2.5 小时)	5
2.1 单层直接法 (3 分)	5
2.2 多层直接法 (2 分)	5
2.3 * 延伸讨论	6
3 * 使用光流计算视差 (2 分, 约 1 小时)	8

1 LK 光流 (5 分, 约 3 小时)

1.1 光流文献综述 (1 分)

我们课上演示了 Lucas-Kanade 稀疏光流，用 OpenCV 函数实现了光流法追踪特征点。实际上，光流法有很长时间的研究历史，直到现在人们还在尝试用 Deep learning 等方法对光流进行改进 [1][1]。本题将指导你完成基于 Gauss-Newton 的金字塔光流法。首先，请阅读文献 [3]，回答下列问题。

问题

1. 按此文的分类，光流法可分为哪几类？
2. 在 compositional 中，为什么有时候需要做原始图像的 warp？该 warp 有何物理意义？
3. forward 和 inverse 有何差别？

Solution:

1. Forward Additional Image Alignment, Forward Composition Image Alignment, Inverse Composition Image Alignment, and Inverse Additional Image Alignment
2. the compositional approach iteratively solves for an incremental warp $W(x; \Delta p)$ rather than an additive update to the parameters Δp . Physically, $W(x; \Delta p)$ means a slightly transformation of original image.
3. The difference between forward and inverse approach is switching the role of the image and the template. Therefore, in inverse approach, the Hessian can be pre-computed and reused in every iteration, and further it can reduce the computation complexity.

1.2 forward-addtive Gauss-Newton 光流的实现 (1 分)

接下来我们来实现最简单的光流，即上文所说的 forward-addtive。我们先考虑单层图像的 LK 光流，然后再推广至金字塔图像。按照教材的习惯，我们把光流法建模成一个非线性优化问题，再使用 Gauss-Newton 法迭代求解。设有图像 1.png, 2.png，我们在 1.png 中提取了 GFTT 角点 [4]，然后希望在 2.png 中追踪这些关键点。设两个图分别为 I_1, I_2 第一张图中提取的点集为 $P = \{p_i\}$ ，其中 $p_i = [x_i, y_i]^T$ 为像素坐标值。考虑第 i 个点，我们希望计算 $\Delta x_i, \Delta y_i$ ，满足：

$$\min_{\Delta x_i, \Delta y_i} \sum_W \|I_1(x_i, y_i) - I_2(x_i + \Delta x_i, y_i + \Delta y_i)\|_2^2 \quad (1)$$

即最小化二者灰度差的平方，其中 \sum_W 表示我们在某个窗口 (Window) 中求和（而不是单个像素，因为问题有两个未知量，单个像素只有一个约束，是欠定的）。实践中，取此 window 为 8×8 大小的小块，即从 $x_i - 4$ 取到 $x_i + 3$, y 坐标亦然。显然，这是一个 forward-addtive 的光流，而上述最小二乘问题可以用 Gauss-Newton 迭代求解。请回答下列问题，并根据你的回答，实现 optical_flow.cpp 文件中的 OpticalFlowSingleLevel 函数。

1. 从最小二乘角度来看，每个像素的误差怎么定义？
2. 误差相对于自变量的导数如何定义？

下面是有关实现过程中的一些提示：

1. 同上一次作业，你仍然需要去除那些提在图像边界附近的点，不然你的图像块可能越过边界。
2. 该函数称为单层的光流，下面我们要基于这个函数来实现多层的光流。在主函数中，我们对两张图像分别测试单层光流、多层次光流，并与 OpenCV 结果进行对比。作为验证，正向单层光流结果应该如图1所示，它结果不是很好，但大部分还是对的。
3. 在光流中，关键点的坐标值通常是浮点数，但图像数据都是以整数作为下标的。之前我们直接取了浮点数的整数部分，即把小数部分归零。但是在光流中，通常的优化值都在几个像素内变化，所以我们还用浮点数的像素插值。函数 GetPixelValue 为你提供了一个双线性插值方法（这也是常用的图像插值法之一），你可以用它来获得浮点的像素值。

Solution:

For the programming task please refer to folder 1.

$$1. G(p) = I(W(x, p + \Delta p)) - T(x)$$

$$2. \frac{\partial G}{\partial p} = \nabla I \frac{\partial W}{\partial p} = [\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}]$$

1.3 反向法 (1 分)

在你实现了上述算法之后，就会发现，在迭代开始时，Gauss-Newton 的计算依赖于 I_2 在 $(x_i + \Delta x_i, y_i + \Delta y_i)$ 处的梯度信息。然而，角点提取算法仅保证了 $I_1(x_i, y_i)$ 处是角点（可以认为角度点存在明显梯度），但对于 I_2 ，我们并没有办法假设 I_2 在 x_i, y_i 处亦有梯度，从而 Gauss-Newton 并不一定成立。反向的光流法 (inverse) 则做了一个巧妙的技巧，即用 $I_1(x_i, y_i)$ 处的梯度，替换掉原本要计算的 $I_2(x_i + \Delta x_i, y_i + \Delta y_i)$ 的梯度。这样做好处有：

- $I_1(x_i, y_i)$ 是角点，梯度总有意义；
- $I_1(x_i, y_i)$ 处的梯度不随迭代改变，所以只需计算一次，就可以在后续的迭代中一直使用，节省了大量计算时间。

我们为 OpticalFlowSingleLevel 函数添加一个 bool inverse 参数，指定要使用正常的算法还是反向的算法。请你根据上述说明，完成反向的 LK 光流法。

Solution:

For the programming task please refer to folder 1.

1.4 推广至金字塔 (2 分)

通过实验，可以看出光流法通常只能估计几个像素内的误差。如果初始估计不够好，或者图像运动太大，光流法就无法得到有效的估计（不像特征点匹配那样）。但是，使用图像金字塔，可以让光流对图像运动不那么敏感。下面请你使用缩放倍率为 2，共四层的图像金字塔，实现 coarse-to-fine 的 LK 光流。函数在 OpticalFlowMultiLevel 中。

实现完成后，给出你的光流截图（正向、反向、金字塔正向、金字塔反向），可以和 OpenCV 作比较。然后回答下列问题：

1. 所谓 coarse-to-fine 是指怎样的过程？
2. 光流法中的金字塔用途和特征点法中的金字塔有何差别？

提示：你可以使用上面写的单层光流来帮助你实现多层光流。

Solution: For the programming task please refer to folder 1.

1. coarse-to-fine means tracking the optical flow from the top layer to bottom layer in the image pyramid. In the top most image, the features are coarse as the image is downsampled, whereas the bottom image the features are finer as the image resolution is much higher.
2.
 - In optical flow, the image pyramid is aimed at solving the problem that the matching point can't be found when for example the camera is moving too fast.
 - In feature detection, image pyramid is to solve scale invariant problem.

1.5 讨论

现在你已经自己实现了光流，看到了基于金字塔的 LK 光流能够与 OpenCV 达到相似的效果（甚至更好）。根据光流的结果，你可以和上讲一样，计算对极几何来估计相机运动。下面针对本次实验结果，谈谈你对下面问题的看法：

- 我们优化两个图像块的灰度之差真的合理吗？哪些时候不够合理？你有解决办法吗？
- 图像块大小是否有明显差异？取 16×16 和 8×8 的图像块会让结果发生变化吗？

- 金字塔层数对结果有怎样的影响？缩放倍率呢？

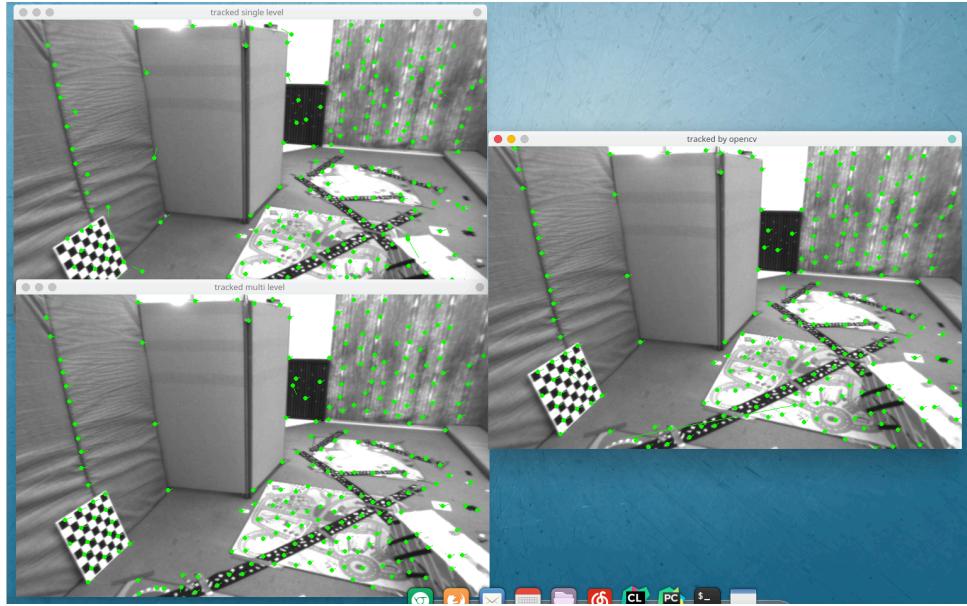


图 1: 光流结果示意图，多层次光流结果应该和 OpenCV 光流类似。

Solution:

For the programming task please refer to folder 1.

- Optical flow approach assumes the observed brightness of any object point is constant over time. However, due to some shadow area and inhomogeneous light condition, this assumption cannot hold true when the camera is moving. To solve this problem, we should calibrate the camera in terms of brightness, to avoid the shadow effect caused by movement and light.
- The bigger the patch is, the robust the tracking performs. The smaller the patch is, the less mis-match points it computes.
- The more layer the pyramid has and the larger the scale is, the easier to track the feature point, but the mismatch of feature points also increase. A good way to fine tune the model is to use more layers in the pyramid and small scale between them.

2 直接法 (5 分, 约 2.5 小时)

2.1 单层直接法 (3 分)

我们说直接法是光流的直观拓展。在光流中，我们估计的是每个像素的平移（在 additive 的情况下）。而在直接法当中，我们最小化光流误差，来估计相机的旋转和平移（以李代数的形式）。现在我们将使用和前一个习题非常相似的做法来实现直接法，请同学体现二者之间的紧密联系。

本习题中，你将使用 Kitti 数据集中的一些图像。给定 left.png 和 disparity.png，我们知道，通过这两个图可以得到 left.png 中任意一点的 3D 信息。现在，请你使用直接法，估计图像 000001.png 至 000005.png 的相机位姿。我们称 left.png 为参考图像 (reference, 简称 ref)，称 000001.png-000005.png 中任意一图为当前图像 (current, 简称 cur)，如图2所示。设待估计的目标为 $\mathbf{T}_{cur,ref}$ ，那么在 ref 中取一组点 p_i ，位姿可以通过最小化下面的目标函数求解：

$$\mathbf{T}_{cur,ref} = \frac{1}{N} \sum_{i=1}^N \sum_{W_i} \|I_{ref}(\pi(\mathbf{p}_i)) - I_{cur}(\pi(\mathbf{T}_{cur,ref}\mathbf{p}_i))\|_2^2 \quad (2)$$

其中 N 为点数， π 函数为针孔相机的投影函数 $\mathbb{R}^3 \rightarrow \mathbb{R}^2$ ， W_i 为第 i 个点周围的小窗口。同光流法，该问题可由 GaussNewton 函数求解。请回答下列问题，然后实现 code/direct_method.cpp 中的 DirectPoseEstimationSingleLayer 函数。

1. 该问题中的误差项是什么？
2. 误差相对于自变量的雅可比维度是多少？如何求解？
3. 窗口可以取多大？是否可以取单个点？

下面是一些实现过程中的提示：

1. 这次我们在参考图像中随机取 1000 个点，而不是取角点。请思考为何不取角点，直接法也能工作。
2. 由于相机运动，参考图像中的点可能在投影之后，跑到后续图像的外部。所以最后的目标函数要对投影在内部的点求平均误差，而不是对所有点求平均。程序中我们以 good 标记出投影在内部的点。
3. 单层直接法的效果不会很好，但是你可以查看每次迭代的目标函数都会下降。

Solution

For the programming task please refer to folder 2.

1. Let ξ is the corresponding Lie algebra of $T_{cur,ref}$, error item is defined as

$$e_i(\xi) = I_{ref}(\mathbf{p}_{1,i}) - I_{cur}(\mathbf{p}_{2,i}) = I_{ref}(\pi(\mathbf{p}_i)) - I_{cur}(\pi(\mathbf{T}_{cur,ref}\mathbf{p}_i)) = I_{ref}(\pi(\mathbf{p}_i)) - I_{cur}(\pi(\exp(\hat{\xi})\mathbf{p}_i))$$

2. According to lecture, $\dim(\mathbf{J}(\hat{\xi})) = 2 \times 6$ and

$$\mathbf{J}(\hat{\xi}) = \begin{bmatrix} \frac{f_x}{Z} & 0 & -\frac{f_x X}{Z^2} & -\frac{f_x X Y}{Z^2} & f_x + \frac{f_x X^2}{Z^2} & -\frac{f_x Y}{Z} \\ 0 & \frac{f_y}{Z} & -\frac{f_y Y}{Z^2} & -f_y - \frac{f_y Y^2}{Z^2} & \frac{f_y X Y}{Z^2} & \frac{f_y X}{Z} \end{bmatrix}$$

3. The patch size depends on tolerant computation time and error about depth information. If the patch is chosen to be a single pixel, the gradient of this pixel will be prone to error.

2.2 多层直接法 (2 分)

下面，类似于光流，我们也可以把直接法以 coarse-to-fine 的过程，拓展至多层金字塔。多层次金字塔的直接法允许图像在发生较大运动时仍能追踪到所有点。下面我们使用缩放倍率为 2 的四层金字塔，实现金字塔上的直接法。请实现 DirectPoseEstimationMultiLayer 函数，下面是一些提示：

1. 在缩放图像时，图像内参也需要跟着变化。那么，例如图像缩小一倍， f_x, f_y, c_x, c_y 应该如何变化？
2. 根据 coarse-to-fine 的过程，上一层图像的位姿估计结果可以作为下一层图像的初始条件。
3. 在调试期间，可以画出每个点在 ref 和 cur 上的投影，看看它们是否对应。若准确对应，则说明位姿估计是准确的。

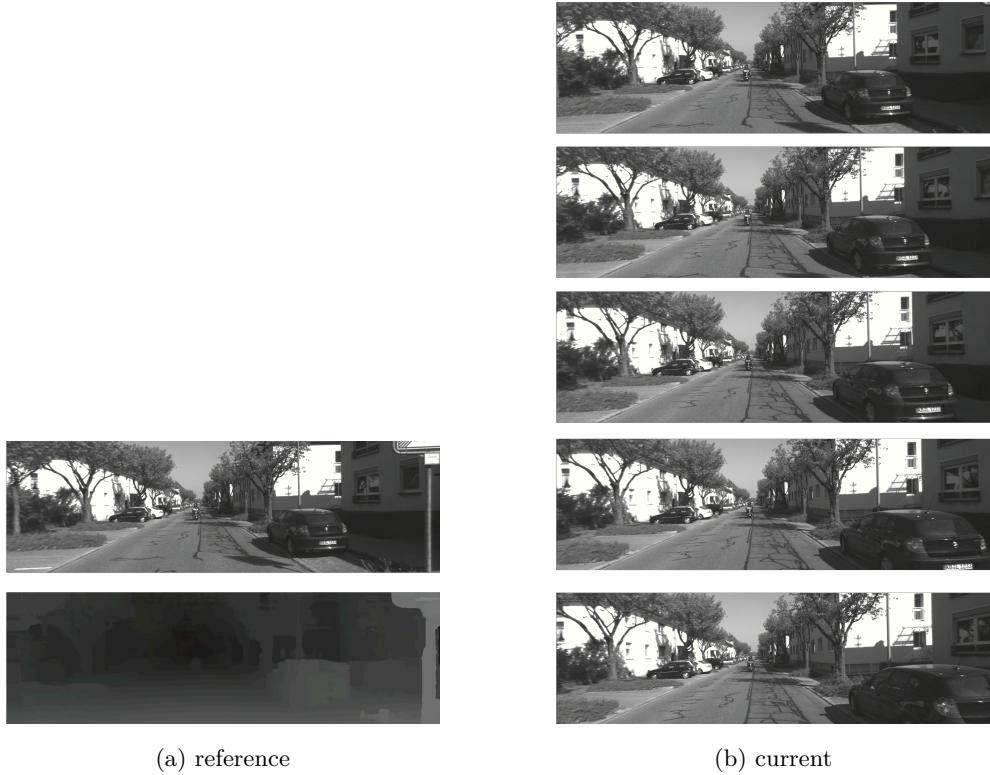


图 2: 本题中待估计位姿的图像

作为验证，图像 000001 和 000005 的位姿平移部分应该接近：

$$\mathbf{t}_1 = [0.005876, -0.01024, -0.725]^T \quad \mathbf{t}_5 = [0.0394, -0.0592, -3.9907]^T \quad (3)$$

可以看出车辆基本是笔直向前开的。

Solution:

For the programming task please refer to folder 2.

2.3 * 延伸讨论

现在你已经实现了金字塔上的 Gauss-Newton 直接法。你可以调整实验当中的一些参数，例如图像点数、每个点周围小块的大小等等。请思考下面问题：

1. 直接法是否可以类似光流，提出 inverse, compositional 的概念？它们有意义吗？
2. 请思考上面算法哪些地方可以缓存或加速？
3. 在上述过程中，我们实际假设了哪两个 patch 不变？
4. 为何可以随机取点？而不用取角点或线上的点？那些不是角点的地方，投影算对了吗？
5. 请总结直接法相对于特征点法的异同与优缺点。

Solution

1. We could have the inverse and composition concept as in optical flow, but it does not make much sense. As the computation complexity won't be reduced significantly in either cases.
2. one can decrease the patch size to accelerate the computation.
3. The grayscale and depth keep constant.
4. Because the computation is based on grayscale value and the camera pose estimation is aimed to

make sure the most selected points have the same greyscale value in both images. However, at corner point, the gradient can be always available.

5. • direct method:

– advantages:

- * avoid computation of descriptor
- * avoid corner point detection, can be used in images without corner point. (eg. only lines on images)
- * can manually choose dense or sparse points to track.

– disadvantages:

- * not convex, prone to local minimum.
- * can be affected by inhomogeneous lighting condition.

• feature based approach

– advantages:

- * robust against lighting condition;
- * robust against high speed camera movement.

– disadvantages:

- * high computation complexity;
- * unstable when the image have too few feature points.

3 * 使用光流计算视差 (2 分, 约 1 小时)

请注意本题为附加题。

在上一题中我们已经实现了金字塔 LK 光流。光流有很多用途，它给出了两个图像中点的对应关系，所以我们可以用光流进行位姿估计，或者计算双目的视差。回忆第四节课的习题中，我们介绍了双目可以通过视差图得出点云，但那时直接给出了视差图，而没有进行视差图的计算。现在，给定图像 left.png, right.png, 请你使用上题的结果，计算 left.png 中的 GFTT 点在 right.png 中的（水平）视差，然后与 disparity.png 进行比较。这样的结果是一个稀疏角点组成的点云。请计算每个角点的水平视差，然后对比视差图比较结果。

本程序不提供代码框架，请你根据之前习题完成此内容。

Solution:

For the programming task please refer to folder 3.

From std output we see the computed disparity is almost the same as the value read from disparity.png.

References

- [1] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, “Flownet: Learning optical flow with convolutional networks,” in Proceedings of the IEEE International Conference on Computer Vision, pp. 2758–2766, 2015.
- [2] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, “Flownet 2.0: Evolution of optical flow estimation with deep networks,” arXiv preprint arXiv:1612.01925, 2016.
- [3] S. Baker and I. Matthews, “Lucas-kanade 20 years on: A unifying framework,” International journal of computer vision, vol. 56, no. 3, pp. 221–255, 2004.
- [4] J. Shi and C. Tomasi, “Good features to track,” in Computer Vision and Pattern Recognition, 1994. Proceedings CVPR’ 94., 1994 IEEE Computer Society Conference on, pp. 593–600, IEEE, 1994.