BundleFusion代码解析

Lingye ···

论文解析参考：blog.csdn.net/fuxingyin...

重要参数

- s_integrationWidth = 320：使用的帧size
- s_maxNumImages = 1200：最大帧数
- s_submapSize = 10：每个localBundle最大帧数
- s_maxNumKeysPerImage = 1024：每帧最大特征点数
- s_colorFilter = false
- s_numSolveFramesBeforeExit = 30
- s_numLocalNonLinIterations=2
- s_numLocalLinIterations = 100
- s_numGlobalNonLinIterations = 3
- s_numGlobalLinIterations = 150
- s_downsampledWidth = 80
- s_maxFrameFixes = 10 //max number of frames reintegrated per frame
- s_RenderMode=1
- s_sensorIdx = 7（StructureSensor）
- s_denseOverlapCheckSubsampleFactor=4

FriedLiver：程序入口

- main：
  - startDepthSensing：调用DX进行渲染，从传感器中读取数据，重建Fusion
- bundlingThreadFunc：多线程
  - 如果已经处理的最后一帧是localBundle的第一帧，则等待上一次solve优化完
  - 如果已经处理的最后一帧是localBundle的最后一帧，则开始新的solve优化（bundlingOptimization）
  - 等待新的输入数据，处理输入processInput，通知DepthSensing

DepthSensing

- OnD3D11FrameRender：DXUTSetCallbackD3D11FrameRender
  - 等待上一次bundlingThread完成，读取数据，然后通知bundlingThread
  - reintegrate
  - 等待bundling完成：sift extraction, sift matching, and key point filtering
  - Reconstruction of current frame
    - integrate：获取当前帧，fusion进TSDF中
  - Render
    - RayCast

CUDAImageManager

- init
  - m_bHasBundlingFrameRdy=false
  - m_currFrame = 0
- process：输入数据预处理，m_currFrame++

- ColorMap：
  - 重采样：根据s_integrationWidth
- DepthMap：
  - erode：腐蚀2次
  - smooth：双边滤波
  - 重采样：根据s_integrationWidth
- 加入m_data

OnlineBundler

- init
  - m_bHasProcessedInputFrame=false
  - m_bExitBundlingThread=false
  - m_lastFrameProcessed = -1
- getCurrentFrame：BundlerInputData获取depthMap和colorMap
  - Color to Intensity：(0.299f*c.x + 0.587f*c.y + 0.114f*c.z) / 255.0f
  - colorFilter：双边滤波bilateral filter
- processInput
  - 获取depth/color data
  - localBundle检测SIFT特征点，缓存数据
  - 如果是localBundle最后一帧，copy localBundle data to optBundle
  - 如果不是localBundle第一帧，localBundle进行SIFT特征点匹配和过滤，计算当前帧的transform
  - 如果是localBundle最后一帧，准备local solve，更新BundlerState，交换localBundle和optBundle
- process：BundlingOptimization
  - 局部优化
    - localBundle optimize
  - processGlobal
    - fuseToGlobal：把localBundle关键帧加入globalBundle中
    - matchAndFilter
  - 全局优化
    - globalBundle optimize

Bundler

- detectFeatures：检测图片SIFT特征点
- storeCachedFrame：缓存数据
- getCurrFrameNumber：当前Bundler中的帧数
- matchAndFilter：匹配和过滤当前帧和前面所有帧的SIFT特征点，计算3D对应点
- optimize
  - CUDA solve
  - 验证VerifyTrajectoryCU

RGBDSensor

- init
  - m_bIsReceivingFrames=true

SIFTImageManager

- createSIFTImageGPU：创建每帧图片的SIFT特征点数组
- finalizeSIFTImageGPU：m_currentImage++

CUDASolverBundling

- **公式（dense）：**
  - $E_d = n_i^T(\pi^{-1}(D(\pi(T_i^{-1}T_j d_j))) - T_i^{-1}T_j d_j)$

- $E_c = I_j(\pi(d_j)) - I_i(\pi(T_i^{-1}T_j d_j))$
- buildVariablesToCorrespondencesTable: 判断每组pair的匹配特征点数是否超过每帧最大数目，是则置为无效
- FindImageImageCorr_Kernel：寻找合法的帧pair（i，j），j是src，i是target，过滤角度相差过大和对应点数太少的pair，只计算下采样区域
- FindDenseCorrespondences_Kernel：寻找每个pair的对应点数
- WeightDenseCorrespondences_Kernel：计算每个pair的权重，过滤对应点数过少的pair
- computeJacobianBlockRow_i：计算E_d对Ti的偏导数，
  - evalLie_derivI：李代数求导
- addToLocalSystem：计算该帧pair的JtJ和Jtr，每帧pair所有对应点的值会加起来，JtJ只计算下半区
  - $J = [J_i, J_j]$
  - $J^T J = [[J_i^T J_i, J_i^T J_j], [J_j^T J_i, J_j^T J_i]]$
  - $J_i^T J_j$ 和 $J_j^T J_i$ 填充在同一个矩阵块的上下两个半区
  - $J^T r = [J_i^T r, J_j^T r]$
- BuildDenseSystem_Kernel
  - 计算当前对应点是否valid
  - 计算res
  - 计算当前对应点的Jacobian
  - 计算每帧pair所有对应点总的JtJ和Jtr
- BuildDenseSystem
  - maxDenseImPairs = input.numberOfImages * (input.numberOfImages - 1) / 2：pari与（i，j）顺序无关
  - usePairwise=true
  - 寻找帧对pair
  - 计算每个pair合法的点对数
  - 计算每个pair的depth权重
  - 计算每个pair的JtJ和Jtr
  - 填充每个pair的JtJ的上半区
- evalMinusJTFDevice
  - 计算sparse energy的JtJ和Jtr(JtF)
  - 计算sparse+dense的-JtF（b）
  - 计算Preconditioner，只使用sparse的JtJ
- PCGInit_Kernel1
  - 计算sparse+dense的b，Preconditioner(M_inv)
  - 计算r=b-Ax_0，x_0=0
  - 计算p=z=M_inv*r
  - 计算r_t*z
  - Ap=0
- PCGInit_Kernel2：保存old r_t*z
- PCGStep_Kernel0：计算每帧的sparse Jp
- PCGStep_Kernel1a：计算每帧的sparse JtJp (Ap)，与dense共用一个Ap
- PCGStep_Kernel_Dense：计算每帧的dense JtJp (Ap)
  - applyJTJDenseDevice：JtJ分块和p相乘
- PCGStep_Kernel1b：计算所有帧的p_t*Ap和，存在d_scanAlpha[0]
- PCGStep_Kernel2
  - 计算alpha
  - 更新x_delta
  - 计算新的r
  - 计算新的z
  - 计算新的r_t*z，存在d_scanAlpha[1]
- PCGStep_Kernel3
  - 计算beta
  - 更新旧的r_t*z
  - 计算新的p
  - Ap=0
  - 如果循环结束，更新pose
- PCGIteration：PCG循环体
  - 参考：en.wikipedia.org/wiki/C...

- solveBundlingStub
  - 对于每个循环
    - dense：计算每个帧pair的JtJ和Jtr
    - 初始化PCG所需变量
    - 多次PCGIteration
    - 判断是否提前收敛

（待更新）

编辑于 2019-10-15

同时定位和地图构建（SLAM） 三维视觉

文章被以下专栏收录

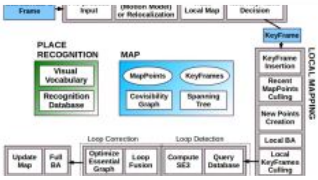
3D视觉感知 进入专栏

推荐阅读

MSCKF那些事（十二）开源算法Open VINS试用

1. Open VINS简介Open VINS是Huang Guoquan老师团队在2019年8月份开源的一套基于MSCKF的VINS算法，黄老师曾是Tango项目的核心成员，在MSCKF这块非常的权威。Github地址:…

紫薯萝卜 发表于VIO/V...



双目ORB-SLAM2代码个人总结（一）

贤鱼卓君 发表于贤鱼从零开...

NLS 问题之(robust co

学习SLAM其在状态估计领题是非常关键中的后端优化非线性最小二题有很多种求

daoju...

1 条评论 ⇌ 切换为时间排序

写下你的评论...

北辰的夏天 26 天前

厉害

赞

▲ 赞同 1 ▼ 1 条评论 分享 收藏 …