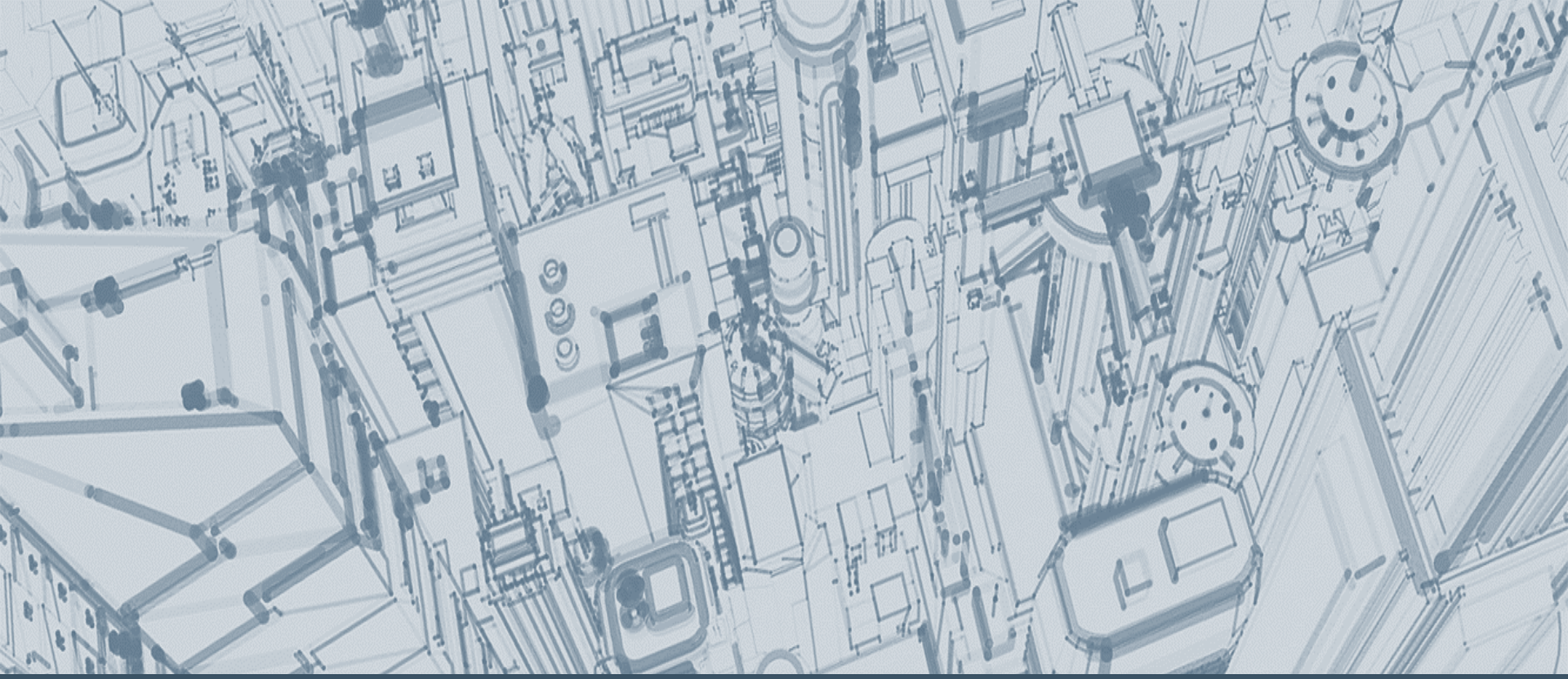


HLS Final: Dense Pose Refinement Pose Engine Hardware

Team 4: Hua-Yang Weng, Ke-Han Li*, Hsin-Yu Chen*

Github: https://github.com/Yuoto/HLS-Final-DPR

Overview

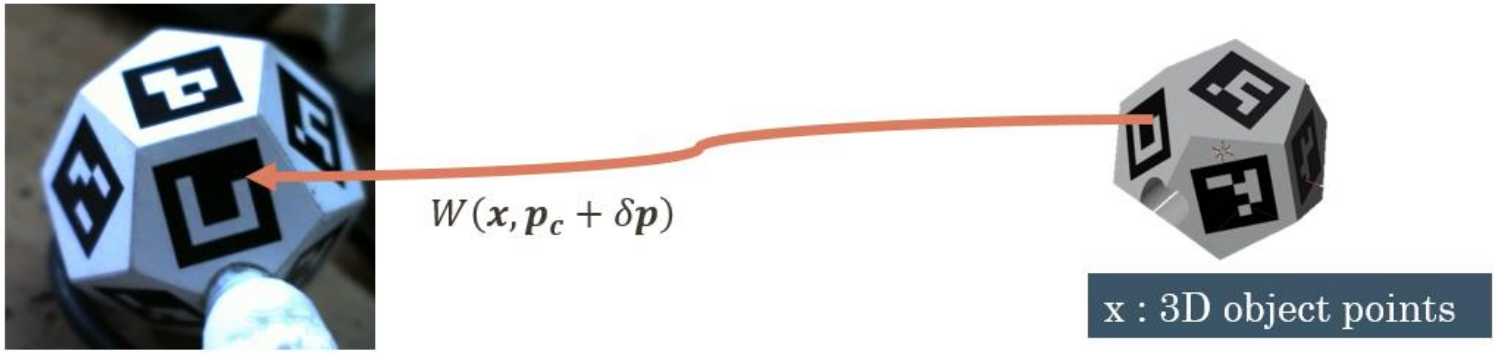
- Direct Method Pose Optimization Problem
- Original Host Code
- Objective: Target to achieve
- System Diagram
- Kernel Code + Optimization
- Evaluation
- Application Demo
- Summary
- Future Work
- Appendix

Overview

- Direct Method Pose Optimization Problem
- Original Host Code
- Objective: Target to achieve
- System Diagram
- Kernel Code + Optimization
- Evaluation
- Application Demo
- Summary
- Future Work
- Appendix

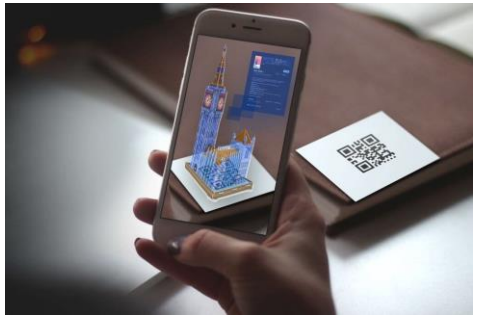
Direct Method Pose Optimization Problem (1/3)

- Find 6DoF(**D**egree **o**f **F**reedom) pose of objects in an image

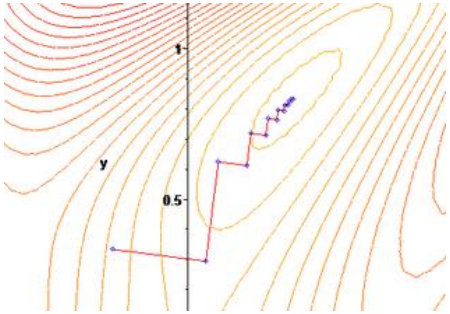


$$F_{FA}(\delta p) = \sum_{x} \left(I\left(W(\boldsymbol{x}, \boldsymbol{p_c} + \delta\boldsymbol{p})\right) - O(\boldsymbol{x}) \right)^2$$

- Application: Mix Reality in Metaverse

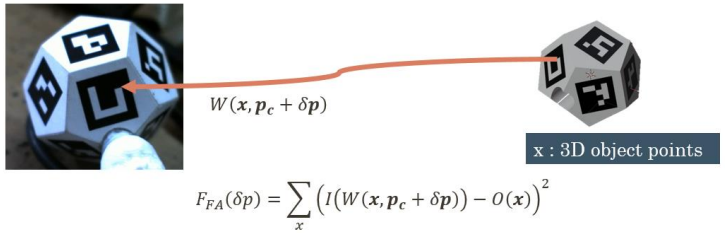


Direct Method Pose Optimization Problem (2/3)



Goal: Optimizing pose by aligning input frame image

$$F_{FA}(\delta p) = \sum_{x} \left(I\left(W(\boldsymbol{x}, \boldsymbol{p_c} + \delta \boldsymbol{p})\right) - O(\boldsymbol{x}) \right)^2$$



Solution: Gauss-Newton or Levenberg-Marquart

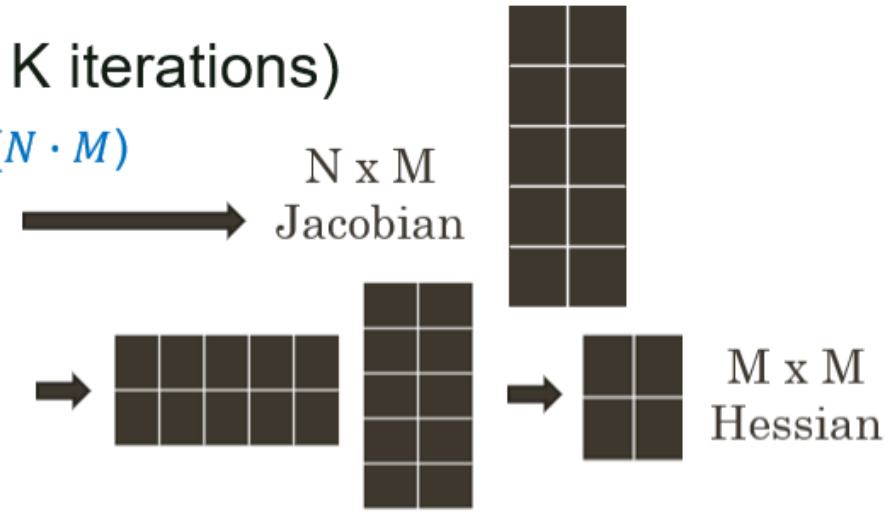
(Traditional Non-linear solvers)

- Calculate derivative with respect to 6DoF pose p ∈ R^6
  - Calculate Jacobian and update

$$J^i(\mathbf{x_c}, \theta\boldsymbol{\xi}_{6\times1})_{1\times6} = \begin{bmatrix} \frac{\partial \mathrm{I}^i}{\partial x_c^i} & \frac{\partial \mathrm{I}^i}{\partial y_c^i} \end{bmatrix} \begin{bmatrix} \frac{f_x}{z_c^i} & 0 & -\frac{f_x x_c^i}{(z_c^i)^2} \\ 0 & \frac{f_y}{z_c^i} & -\frac{f_y y_c^i}{(z_c^i)^2} \end{bmatrix} \begin{bmatrix} 0 & z_c^i & -y_c^i & 1 & 0 & 0 \\ -z_c^i & 0 & x_c^i & 0 & 1 & 0 \\ y_c^i & -x_c^i & 0 & 0 & 0 & 1 \end{bmatrix}$$

Direct Method Pose Optimization Problem (3/3)

- While loop (N points, M vars, K iterations)
  - Cal Xpixel/Xpoint with current pose $O(N \cdot M)$
    - Sampling gradient (bilinear sampling)
    - Cal residual , Jacobian (matrix forming)
  - Cal Hessian $O(N \cdot M^2)$, Jres $O(N \cdot M)$
    - matmul: very large, tall matrix
  - Solve Hx=Jres
    - Cholesky $O(M^3)$, back-substitution $O(M^2)$
  - Line search (L iterations)
    - Cal residual $O(N \cdot M)$



For N=10000, M=6, K=5, L=5

Latency Approximate: K* (N*M+ N*M^2 + N*M + M^3 + M^2+ L*N*M) ~ 3.9M cycles

(Hardware unroll with factor M + partial dataflow)

Latency Approximate: K* (N+ M^3 + M^2+L*N) ~ 0.3M cycles, 3ms @100MHz

Overview

- Direct Method Pose Optimization Problem
- Original Host Code
- Objective: Target to achieve
- System Diagram
- Kernel Code + Optimization
- Evaluation
- Application Demo
- Summary
- Future Work
- Appendix

Original Host Code (1/3)

1. Not supported data type (Due to dynamic allocating)

(**Eigen** -> C++ template library for linear algebra, equivalent to numpy in python)

```
void DodecaSystemTracker::denseAlignment(const cv::Mat& normalizedImage,
    const Region& validRegion,
    const dst::Vec& pixel3D,
    const dst::MatX3& point3D,
    int methodSelection,
    double epsilonRot,
    double epsilonTra,
    int maxIter,
    dst::Mat3* R,
    dst::Vec3* t)
{
    dst::Vec6 p;

    p << Transformation::FromRotationMatirxToAxisAngle(*R), * t;
    dst::Vec6 deltaP = dst::Vec6::Constant(DBL_MAX);
    int iter = 0;
    int num = pixel3D.size();
    dst::Mat34 Rt;
    Rt << *R, * t;
    dst::Vec warpedI(num);
    dst::Vec warpedIu(num);
    dst::Vec warpedIv(num);
```

Need to rewrite in array or pointer!

```
namespace dst
{
    typedef unsigned char byte;

    typedef Eigen::VectorXd Vec;
    typedef Eigen::MatrixXd Mat;

    typedef Eigen::Vector2d Vec2;
    typedef Eigen::Vector3d Vec3;
    typedef Eigen::Vector4d Vec4;
    typedef Eigen::Matrix<double, 6, 1> Vec6;
    typedef Eigen::Matrix<double, 7, 1> Vec7;

    typedef Eigen::RowVector2d RVec2;
    typedef Eigen::RowVector3d RVec3;
    typedef Eigen::RowVector4d RVec4;
    typedef Eigen::Matrix<double, 1, 6> RVec6;
    typedef Eigen::Matrix<double, 1, 7> RVec7;
```

Original Host Code (2/3)

2. Not supported eigen relative high-level-functions

(Eigen class array-wise operation)

Need to rewrite the detail implementations!

```
// --- Step 1: Compute Jfa ---
dst::MatX3 point2D = (point3D * R->transpose()).rowwise() + t->transpose()) * _inMat.transpose();
point2D.leftCols(2).array().colwise() *= 1. / point2D.col(2).array();


    // --- Step 2: Compute H ---
    dst::Mat6 H = J.transpose() * J; // [6, 6] = [6, N] @ [N, 6]

    // --- Step 3: Compute delta p ---
    dst::Vec E = pixel3D - warpedI;  // [N, ]
    dst::Vec6 JtE = J.transpose() * E; // [6, 1]
    deltaP = H.inverse() * JtE; // [6, 1]
```

Original Host Code (3/3)

3. Hardware unfriendly coding style

(Many Dividors, multipliers, adders)

```
double rxcl1_l4 = rx * cl1_l4;
double rycl1_l4 = ry * cl1_l4;
double rzcl1_l4 = rz * cl1_l4;
(*JRr)(0, 0) = -(sl * ry2rz2 * rx_l3) - (2 * ry2rz2 * rxcl1_l4);
(*JRr)(0, 1) = (2 * cl1 * ry_l2) - (sl * ry2rz2 * ry_l3)
    - (2 * ry2rz2 * rycl1_l4);
(*JRr)(0, 2) = (2 * cl1 * rz_l2) - (sl * ry2rz2 * rz_l3)
    - (2 * ry2rz2 * rzcl1_l4);
(*JRr)(1, 0) = (rzsl * rx_l3) - (rzcl * rx_l2) - (cl1 * ry_l2)
    + (rx * rysl * rx_l3) + (2 * rx * ry * rxcl1_l4);
(*JRr)(1, 1) = (rzsl * ry_l3) - (rzcl * ry_l2) - (cl1 * rx_l2)
    + (rx * rysl * ry_l3) + (2 * rx * ry * rycl1_l4);
(*JRr)(1, 2) = (rzsl * rz_l3) - (rzcl * rz_l2) - sl_l
    + (rx * rysl * rz_l3) + (2 * rx * ry * rzcl1_l4);
(*JRr)(2, 0) = (rycl * rx_l2) - (cl1 * rz_l2) - (rysl * rx_l3)
    + (rx * rzsl * rx_l3) + (2 * rx * rz * rxcl1_l4);
(*JRr)(2, 1) = sl_l + (rycl * ry_l2) - (rysl * ry_l3)
    + (rx * rzsl * ry_l3) + (2 * rx * rz * rycl1_l4);
(*JRr)(2, 2) = (rycl * rz_l2) - (cl1 * rx_l2) - (rysl * rz_l3)
    + (rx * rzsl * rz_l3) + (2 * rx * rz * rzcl1_l4);
(*JRr)(3, 0) = (rzcl * rx_l2) - (cl1 * ry_l2) - (rzsl * rx_l3)
    + (rx * rysl * rx_l3) + (2 * rx * ry * rxcl1_l4);
(*JRr)(3, 1) = (rzcl * ry_l2) - (cl1 * rx_l2) - (rzsl * ry_l3)
    + (rx * rysl * ry_l3) + (2 * rx * ry * rycl1_l4);
```

See later kernel code opt!

Overview

- Direct Method Pose Optimization Problem
- Original Host Code
- Objective: Target to achieve
- System Diagram
- Kernel Code + Optimization
- Evaluation
- Application Demo
- Summary
- Future Work
- Appendix

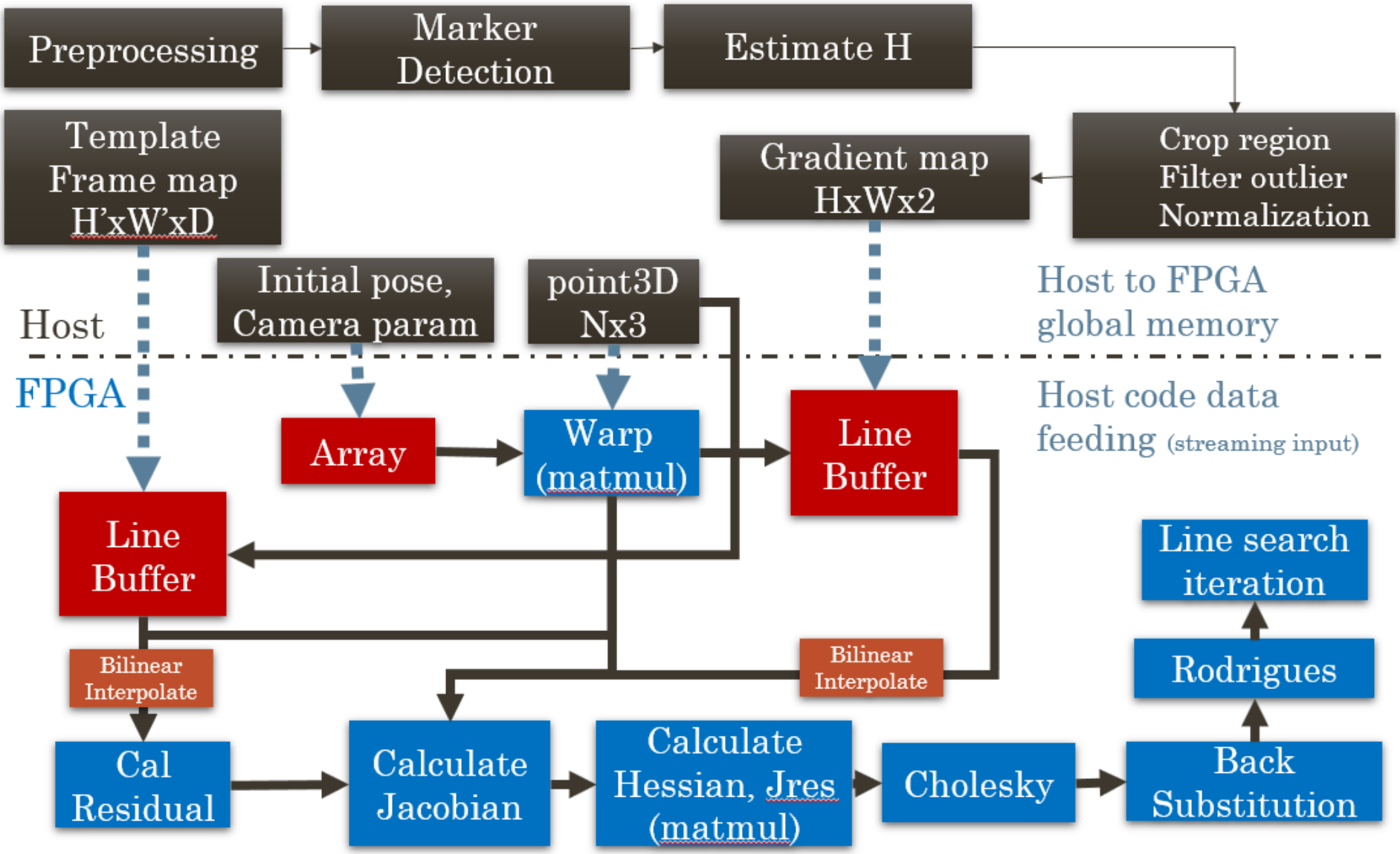
Objective: Target to achieve

- [ ] Given an original C++ code, find the bottleneck and optimize it with kernel code
- [ ] FPS: 74 fps @ 300MHz (N = 10000, M = 6)

Platform to implement:

- [ ] OpenCL & U50 system
- [ ] Software Emulation Pass Testbench
- [ ] Hardware Pass Testbench

System Diagram



Overview

- Direct Method Pose Optimization Problem
- Original Host Code
- Objective: Target to achieve
- System Diagram
- Kernel Code + Optimization
- Evaluation
- Application Demo
- Summary
- Future Work
- Appendix

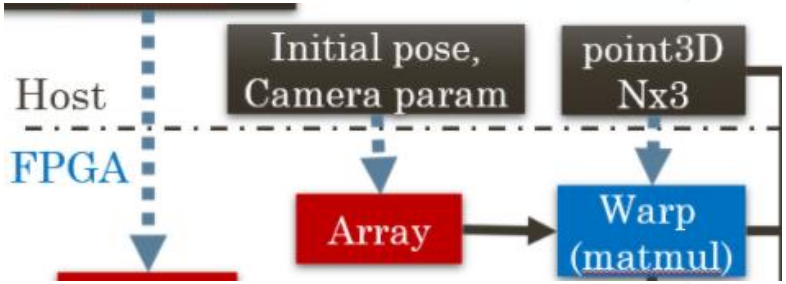
Kernel Code

1. calJacobianMseHLS
2. calHessianJres
3. cholesky_HLS
4. backSubstitutionHLS
5. meanSquaredErrorHLS
6. computeRrodriguesHLS

1. calJacobianMseHLS (1/4)

- Mainly contains a N (~10000) iteration for loop
  - Warp the input points (Matmul)

```
//// fetch SIMD points at once (However, memory is the bottleneck for performing SIMD parallel)
for (k = 0; k < N; k++) {

    //read data from stream
    //inPoint3D_x = point3D_x.read();
    //inPoint3D_y = point3D_y.read();
    //inPoint3D_z = point3D_z.read();
    //inMarkerPixelColor = marker_pixel_color.read();
    inPoint3D_x = point3D_x_buffer[k];
    inPoint3D_y = point3D_y_buffer[k];
    inPoint3D_z = point3D_z_buffer[k];
    inMarkerPixelColor = marker_pixel_color_buffer[k];

    //compute camera coordinate
    camCoord_x = (poseMat[0][0]) * inPoint3D_x + (poseMat[0][1]) * inPoint3D_y + (poseMat[0][2]) * inPoint3D_z
    camCoord_y = (poseMat[1][0]) * inPoint3D_x + (poseMat[1][1]) * inPoint3D_y + (poseMat[1][2]) * inPoint3D_z
    camCoord_z = (poseMat[2][0]) * inPoint3D_x + (poseMat[2][1]) * inPoint3D_y + (poseMat[2][2]) * inPoint3D_z

    //compute pixel coordinate
    camCoord_z_inv = 1. / camCoord_z;
    camCoord_x_div_z = camCoord_x * camCoord_z_inv;
    camCoord_y_div_z = camCoord_y * camCoord_z_inv;


    //pixelCoord_x = int((cameraParam[0] * camCoord_x_div_z + cameraParam[2] - anchor_x)[0]);
    //pixelCoord_y = int((cameraParam[1] * camCoord_y_div_z + cameraParam[3] - anchor_y)[0]);
    pixelCoord_x = int(cameraParam[0] * camCoord_x_div_z + cameraParam[2] - anchor_x);
    pixelCoord_y = int(cameraParam[1] * camCoord_y_div_z + cameraParam[3] - anchor_y);
```



1. calJacobianMseHLS (2/4)

- For each iteration, calculate the access address

- Random access
  - Typically, use a Ping-Pong buffer to store partial array
  - The address might not be loaded -> must guarantee the address generation

However, since we are iterating many times, we store the whole array.

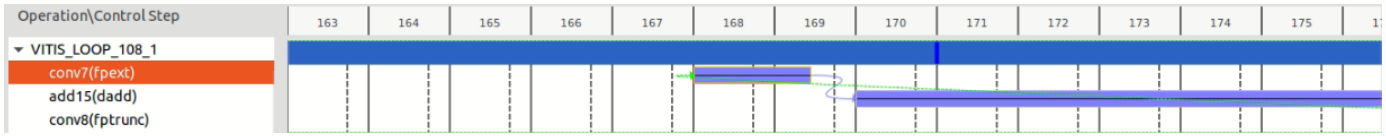
(Otherwise, the host should feed the same data many times & sync)

```
if (pixelCoord_x < 0 || pixelCoord_x > MAX_COLS || pixelCoord_y < 0 || pixelCoord_y > STORE_LINES) {
    frameInterpVal = 1.;
    gxInterpVal = 1.;
    gyInterpVal = 1.;
}
else {
    frameInterpVal = frame_buffer[pixelCoord_y][pixelCoord_x];
    gxInterpVal = gradient_x_buffer[pixelCoord_y][pixelCoord_x];
    gyInterpVal = gradient_y_buffer[pixelCoord_y][pixelCoord_x];
}
```

1.calJacobianMseHLS (3/4)

- Calculate each Jacobian term and stream out

```
J1_tmp = tmp33[0] * gxfxCamCoord_z_inv + tmp33[1] * gyfyCamCoord_z_inv - tmp33[2] * (gxfxCamCoord_z_inv * camCoord_x_div_z + gy
J2_tmp = tmp33[3] * gxfxCamCoord_z_inv + tmp33[4] * gyfyCamCoord_z_inv - tmp33[5] * (gxfxCamCoord_z_inv * camCoord_x_div_z + gy
J3_tmp = tmp33[6] * gxfxCamCoord_z_inv + tmp33[7] * gyfyCamCoord_z_inv - tmp33[8] * (gxfxCamCoord_z_inv * camCoord_x_div_z + gy
//J1_tmp = tmp33[0] * gxfxCamCoord_z_inv[0] + tmp33[1] * gyfyCamCoord_z_inv[0] - tmp33[2] * (gxfxCamCoord_z_inv * camCoord_x_di
//J2_tmp = tmp33[3] * gxfxCamCoord_z_inv[0] + tmp33[4] * gyfyCamCoord_z_inv[0] - tmp33[5] * (gxfxCamCoord_z_inv * camCoord_x_di
//J3_tmp = tmp33[6] * gxfxCamCoord_z_inv[0] + tmp33[7] * gyfyCamCoord_z_inv[0] - tmp33[8] * (gxfxCamCoord_z_inv * camCoord_x_di

//J1_tmp = -1. * gxfx * camCoord_x_div_z * camCoord_y_div_z - gyfy * (1 + camCoord_y_div_z * camCoord_y_div_z);
//J2_tmp = gxfx * (1 + camCoord_x_div_z * camCoord_x_div_z) + gyfy * camCoord_x_div_z * camCoord_y_div_z;
//J3_tmp = -1. * gxfx * camCoord_y_div_z + gyfy * camCoord_x_div_z;

J4_tmp = gxfx * camCoord_z_inv;
J5_tmp = gyfy * camCoord_z_inv;
J6_tmp = -1. * (gxfx * camCoord_x_div_z + gyfy * camCoord_y_div_z) * camCoord_z_inv;
//if (k < 20)
//    std::cout << "(J1, J2, J3, J4, J5, J6) point " << k << " : (" <<J1_tmp[0] << "," << J2_tmp[0] << "," << J3_tmp[0] << ","
J1 << J1_tmp;
J2 << J2_tmp;
J3 << J3_tmp;
J4 << J4_tmp;
J5 << J5_tmp;
J6 << J6_tmp;
residual_tmp = inMarkerPixelColor - frameInterpVal;
residual << residual_tmp;
//residual_sum += pow(residual_tmp[0],2);
residual_sum += pow(residual_tmp, 2);
```



RRRMMMMMMMAAAAAAA → tmp_i[0],
RRRMMMMMMMAAAAAAA → tmp_i[1]
RRRMMMMMMMAAAAAAA → tmp_i[2]
RRRMMMMMMMAAAAAAA → tmp_i[3]
RRRMMMMMMMAAAAAAA → tmp_i[4]
RRRMMMMMMMAAAAAAA → tmp_i[5]
II=6 → RRRMMMMMMMAAAAAAA → tmp_i[6

Modules & Loops	Issue Type	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT
calJacobianMseHLS_float_1_300_400_s	II Violation	-0.24	120209	4.010E5	-	120209	-	no	0	1	6434	3681
pow_generic_double_s		-	78	260.000	-	1	-	yes	5	36	12967	9724
VITIS_LOOP_108_1	II Violation	-	120152	4.000E5	165	12	10000	yes	-	-	-	-

1. calJacobianMseHLS (4/4)

- Fixed II problem

Before: Cycles: 120209, II = 12, FF: 6434, LUT: 3681

Modules & Loops	Issue Type	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT
calJacobianMseHLS_float_1_300_400_s	II Violation	-0.24	120209	4.010E5	-	120209	-	no	0	1	6434	3681
pow_generic_double_s		-	78	260.000	-	1	-	yes	5	36	12967	9724
VITIS_LOOP_108_1	II Violation	-	120152	4.000E5	165	12	10000	yes	-	-	-	-

After: Cycles: 10280, II = 1, FF: 7684, LUT: 3394

Modules & Loops	Issue Type	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
calJacobianMseHLS_float_1_300_400_s		-0.20	10280	3.426E4	-	10280	-	no	0	1	7684	3394	0
pow_generic_double_s		-	78	260.000	-	1	-	yes	5	36	12967	9724	0
Loop 1		-	16	53.328	1	-	16	no	-	-	-	-	-
VITIS_LOOP_108_1		-	10164	3.388E4	166	1	10000	yes	-	-	-	-	-
VITIS_LOOP_217_4		-	16	53.328	10	1	8	yes	-	-	-	-	-
VITIS_LOOP_219_5		-	11	36.663	9	1	4	yes	-	-	-	-	-
VITIS_LOOP_221_6		-	9	29.997	9	1	2	yes	-	-	-	-	-

FF: x 1.2 increase

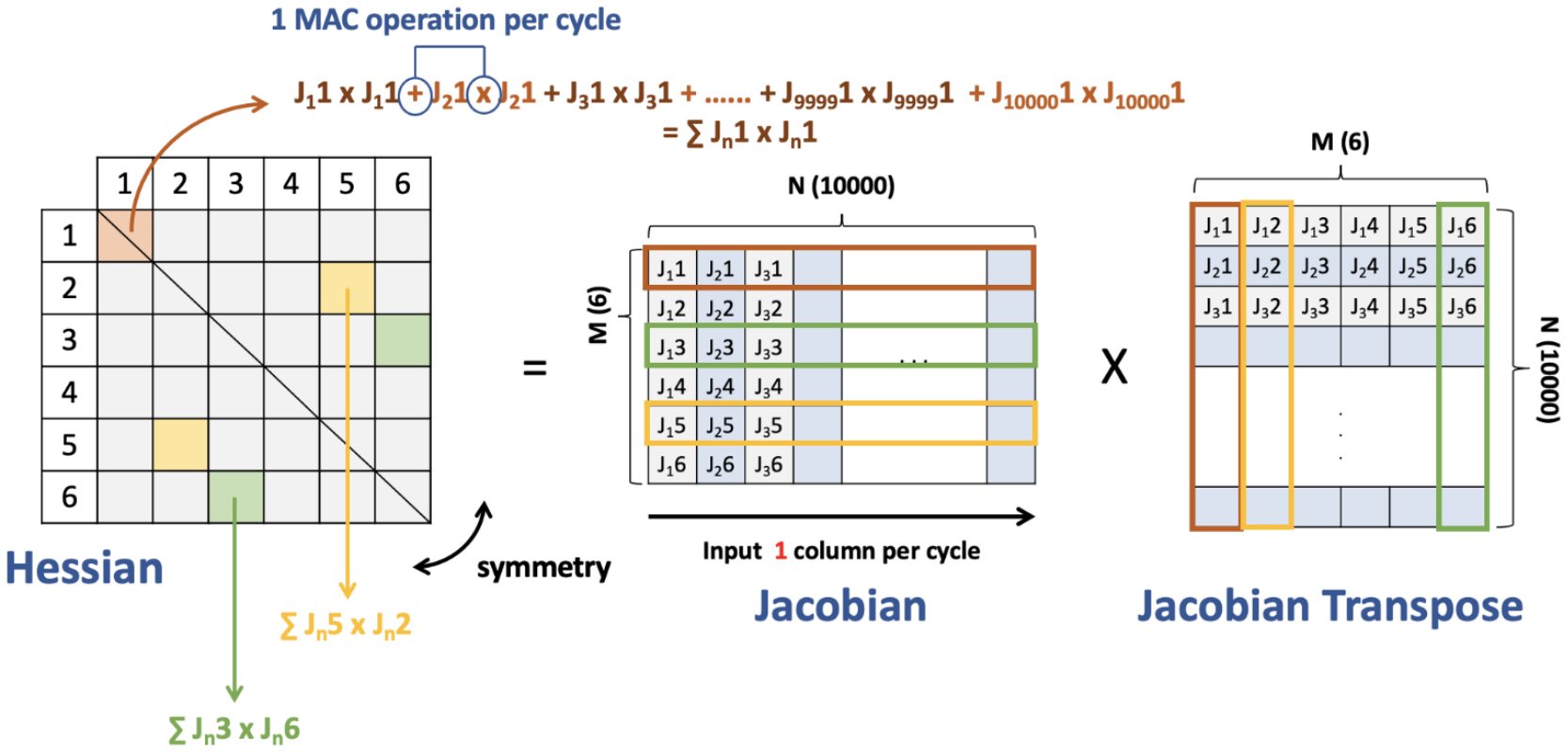
LUT: x 0.92 decrease

Cycles: x 12 decrease

```
T residual_sum_tmp[16] = {0}, residual_sum_tmp1[8], residual_sum_tmp2[4], residual_sum_tmp3[2];
        residual_sum_tmp[k%16] += pow(residual_tmp, 2);
    }
    for (int add=0;add<8;add++)
        residual_sum_tmp1[add] = residual_sum_tmp[add] + residual_sum_tmp[add+8];
    for (int add=0;add<4;add++)
        residual_sum_tmp2[add] = residual_sum_tmp1[add] + residual_sum_tmp1[add+4];
    for (int add=0;add<2;add++)
        residual_sum_tmp3[add] = residual_sum_tmp2[add] + residual_sum_tmp2[add+2];
    residual_sum = (residual_sum_tmp3[0] + residual_sum_tmp3[1]) * num_inv;
```

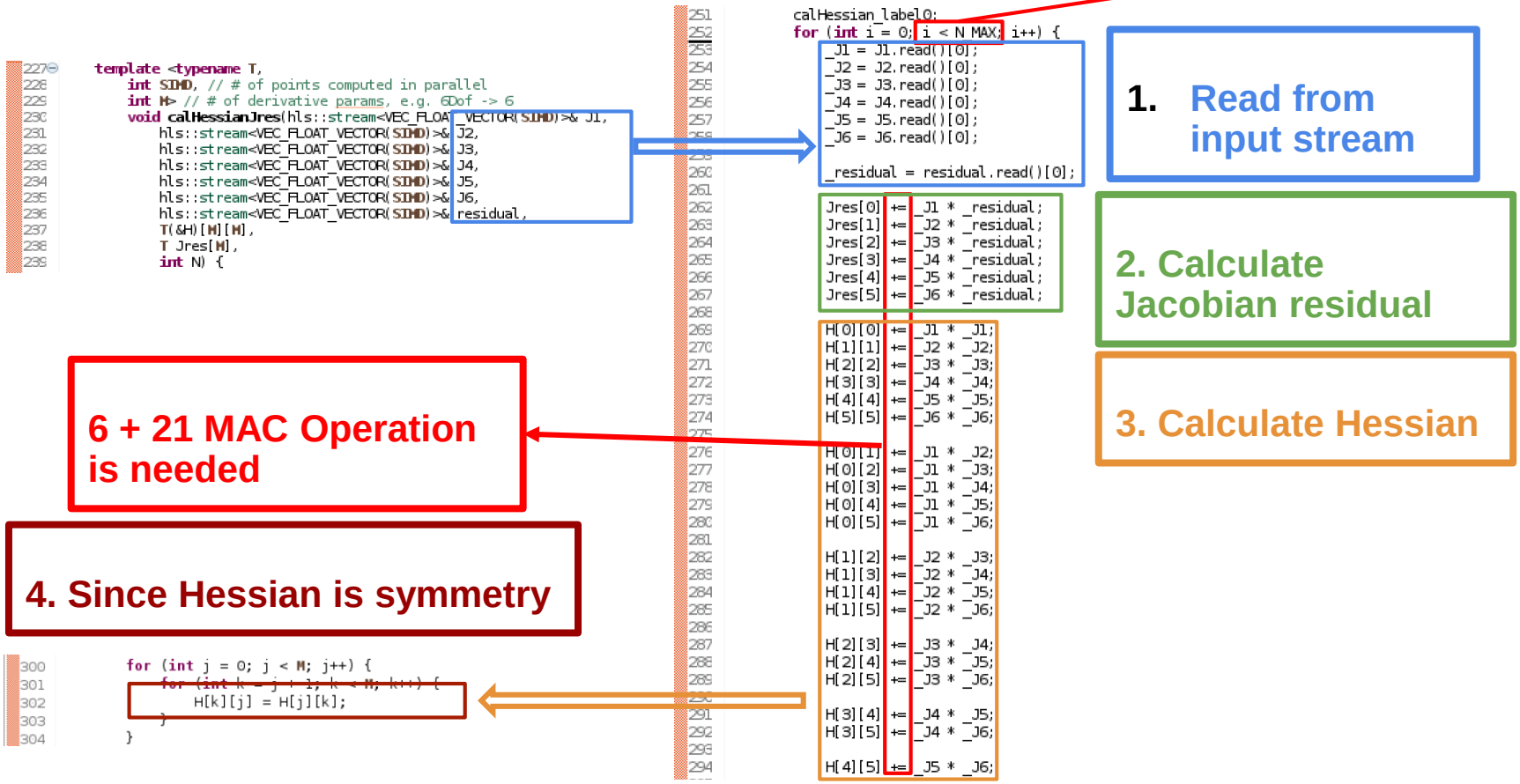
2. calHessianJres(1/3)

- Functionality



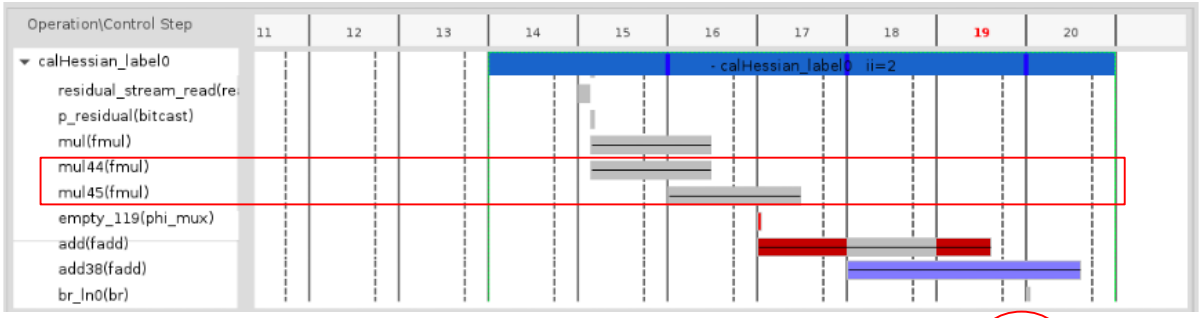
2. calHessianJres(2/3)

- Calculate both Hessian and Jacobian residual



2. calHessianJres(3/3)

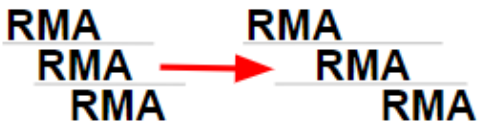
- Original Code: **II = 2 + timing violation**
- **“+=”: Data Dependency**



Modules & Loops	Issue Type	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count
denseAlignment_api		-5.00	296871	3.651E6	-	296872	-
denseAlignmentHLS_float_1_300_400_s		-5.00	296870	3.651E6	-	296870	-
calJacobianMseHLS_float_1_300_400_s		-	10061	1.010E5	-	10061	-
meanSquaredErrorHLS_float_1_300_400_s		-	598	5.980E3	-	598	-
calHessianJres_float_1_6_s	II&Timing Violation	-5.00	20117	2.470E5	-	20117	-
VITIS_LOOP_241_1		-	48	590.000	8	-	6
VITIS_LOOP_251_3	II&Timing Violation	-	20004	2.460E5	7	2	10000
VITIS_LOOP_299_4		-	40	492.000	8	-	5
backSubstitutionHLS_float_6_s		-0.05	124	1.240E3	-	124	-
VITIS_LOOP_653_1_VITIS_LOOP_654_2		-	120002	1.476E6	4	1	120000
VITIS_LOOP_666_3		-	10000	1.230E5	2	1	10000
VITIS_LOOP_680_4		-	27	332.000	2	1	27
VITIS_LOOP_713_6		-	166815	2.051E6	33363	-	5
VITIS_LOOP_900_13		-	9	111.000	2	1	9
VITIS_LOOP_902_14		-	6	73.788	2	1	6

WAR problem

Schedule:



II = 1 **II = 2**

Solve II Violation

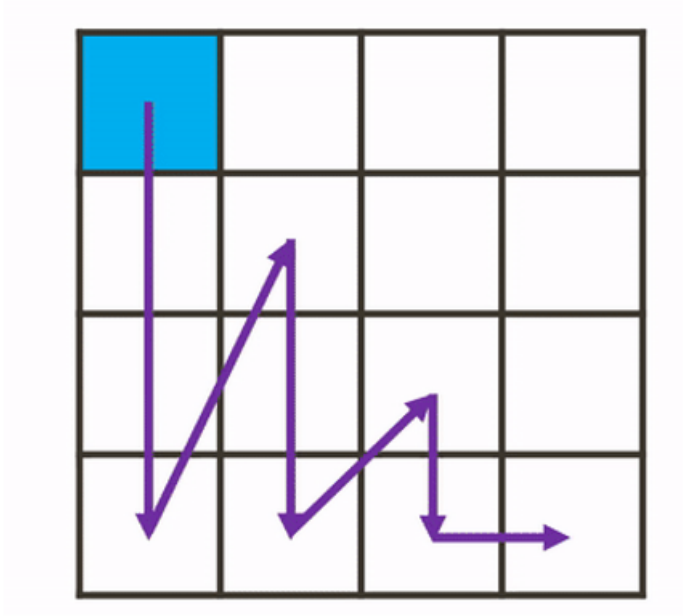
- Use buffer array (size > II, we use 16 here)
- Each element accumulates value every 16 iterations.
- After this 10000 loop, use adder tree to add these 16 elements.

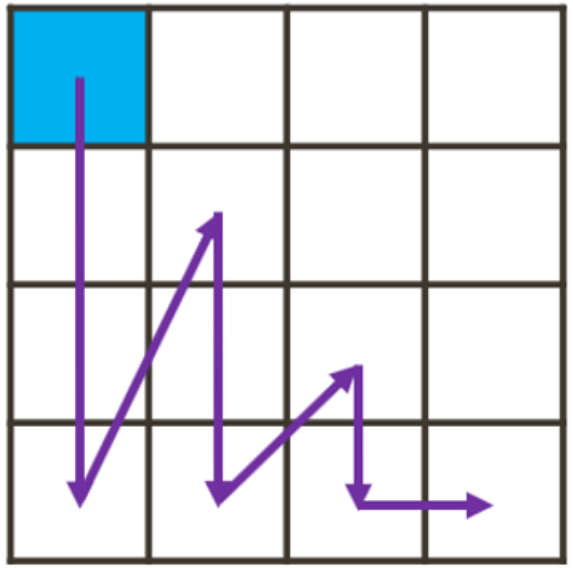

```
temp_Jres0[i%16] += _J1 * _residual;
for (int add=0;add<8;add++){
    temp0_Jres0[add] = temp_Jres0[add] + temp_Jres0[add+8];
}
for (int add=0;add<4;add++){
    temp1_Jres0[add] = temp0_Jres0[add] + temp0_Jres0[add+4];
}
for  (int add=0;add<2;add++){
    temp2_Jres0[add] = temp1_Jres0[add] + temp1_Jres0[add+2];
}
```

Modules & Loops	Issue Type	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count
Loop 19		-	8	80.000	1	-	8
Loop 20		-	8	80.000	1	-	8
Loop 21		-	8	80.000	1	-	8
Loop 22		-	8	80.000	1	-	8
calHessian_label0		-	10004	1.000E5	6	1	10000
VITIS_LOOP_343_3		-	6	60.000	4	1	4
VITIS_LOOP_373_4		-	3	30.000	3	1	2
VITIS_LOOP_429_5		-	40	400.000	8	-	5

3. cholesky_HLS (1/2)

$$\mathbf{A} = \mathbf{L}\mathbf{L}^{\mathbf{T}} = \begin{pmatrix} \mathbf{L}_{11} & 0 & 0 \\ \mathbf{L}_{21} & \mathbf{L}_{22} & 0 \\ \mathbf{L}_{31} & \mathbf{L}_{32} & \mathbf{L}_{33} \end{pmatrix} \begin{pmatrix} \mathbf{L}_{11} & \mathbf{L}_{21} & \mathbf{L}_{31} \\ 0 & \mathbf{L}_{22} & \mathbf{L}_{32} \\ 0 & 0 & \mathbf{L}_{33} \end{pmatrix}$$
$$= \begin{pmatrix} \mathbf{L}_{11}^2 & & (\text{symmetric}) \\ \mathbf{L}_{21}\mathbf{L}_{11} & \mathbf{L}_{21}^2 + \mathbf{L}_{22}^2 & \\ \mathbf{L}_{31}\mathbf{L}_{11} & \mathbf{L}_{31}\mathbf{L}_{21} + \mathbf{L}_{32}\mathbf{L}_{22} & \mathbf{L}_{31}^2 + \mathbf{L}_{32}^2 + \mathbf{L}_{33}^2 \end{pmatrix}$$





$$\mathbf{L}_{j,j} = \sqrt{A_{j,j} - \sum_{k=1}^{j-1} \mathbf{L}_{j,k}^2}$$

$$\mathbf{L}_{i,j} = \frac{1}{\mathbf{L}_{j,j}} \left(A_{i,j} - \sum_{k=1}^{j-1} \mathbf{L}_{i,k}\mathbf{L}_{j,k} \right), \qquad \text{for } i > j$$

Insights:
- For each column data, they can be computed in parallel
- The order of column cannot be changed

3. cholesky_HLS (2/2)

```
template <typename T, int M>
void cholesky_HLS(T dataA[M][M]) {

    T tmp1 = sqrt(dataA[0][0]);

    dataA[0][0] = tmp1;
Loop_first_col:
    for (int i = 1; i < M; i++) {
        dataA[i][0] = dataA[i][0] / tmp1;
    }

Loop_col:
    for (int j = 1; j < M; ++j) {
        T tmp = 0;

    Loop_diag:
        for (int k = 0; k < j; k++) {
            tmp += dataA[j][k] * dataA[j][k];
        }

        dataA[j][j] = sqrt(dataA[j][j] - tmp);

        if (j < M - 1) {
        Loop_row:
            for (int i = j + 1; i < M; ++i) {
                T tmp2 = 0;
            Loop_vec_mul:
                for (int k = 0; k < j; k++) {
                    tmp2 += dataA[i][k] * dataA[j][k];
                }
                dataA[i][j] = (dataA[i][j] - tmp2) / dataA[j][j];
            }
```

II=1 in the inner-most inner product loop

Overall Latency 953 cycles

Minimal to the system,

Hence, do not perform unrolling

Modules & Loops	Issue Type	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT
VITIS_LOOP_1299_3	-	-	-	-	-	-	-	no	-	-	-	-
Loop_first_col	-	-	18	59.994	15	1	5	yes	-	-	-	-
Loop_col	-	-	935	3.116E3	187	-	5	no	-	-	-	-
Loop_diag	-	-	14	46.662	11	1	5	yes	-	-	-	-
Loop_row	-	-	148	493.000	37	-	4	no	-	-	-	-
Loop_vec_mul	-	-	13	43.329	11	1	4	yes	-	-	-	-

4. backSubstitutionHLS

- The matrix equation **Lx = b** can be written as a system of linear equations

$$
\begin{aligned}
\ell_{1,1}x_1 &&&&&&&= b_1\\
\ell_{2,1}x_1 &+ \ell_{2,2}x_2 &&&&&&= b_2\\
\vdots &\quad \vdots &\quad \ddots &&&&&\vdots\\
\ell_{m,1}x_1 &+ \ell_{m,2}x_2 &+ \cdots &+ \ell_{m,m}x_m &&&&= b_m
\end{aligned}
$$

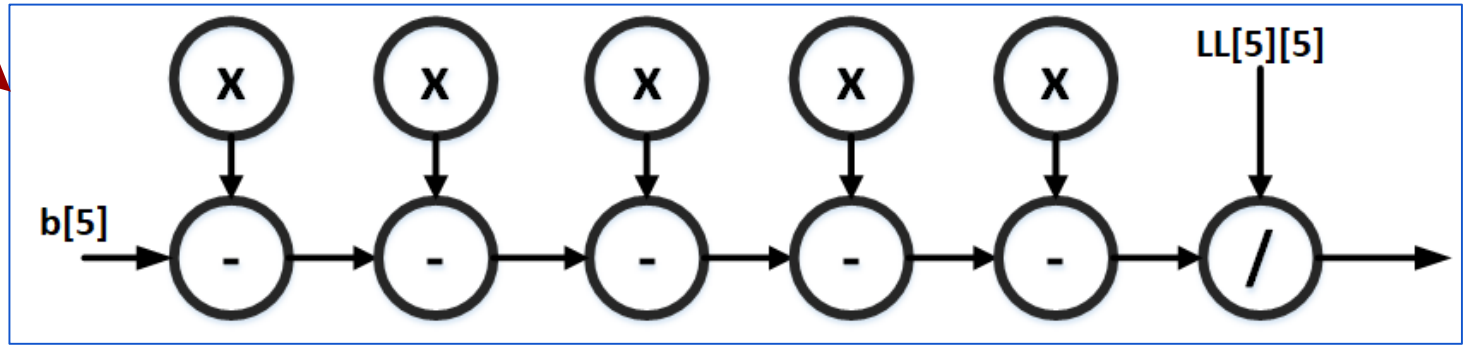
- Observe that the first equation only involves x1,thus we can solve directly.

$$
\begin{aligned}
x_1 &= \frac{b_1}{\ell_{1,1}},\\
x_2 &= \frac{b_2 - \ell_{2,1}x_1}{\ell_{2,2}},\\
&\vdots\\
x_m &= \frac{b_m - \sum_{i=1}^{m-1} \ell_{m,i}x_i}{\ell_{m,m}}.
\end{aligned}
$$

backSubstitution-Baseline

- Write down the HLS code according to the formula directly
  - consume lots of hardware resources
  - latency is very short

```
y[0] = (b[0]) / LL[0][0];
y[1] = (b[1] - LL[1][0] * y[0]) / LL[1][1];
y[2] = (b[2] - LL[2][0] * y[0] - LL[2][1] * y[1]) / LL[2][2];
y[3] = (b[3] - LL[3][0] * y[0] - LL[3][1] * y[1] - LL[3][2] * y[2]) / LL[3][3];
y[4] = (b[4] - LL[4][0] * y[0] - LL[4][1] * y[1] - LL[4][2] * y[2] - LL[4][3] * y[3]) / LL[4][4];
y[5] = (b[5] - LL[5][0] * y[0] - LL[5][1] * y[1] - LL[5][2] * y[2] - LL[5][3] * y[3] - LL[5][4] * y[4]) / LL[5][5];
```

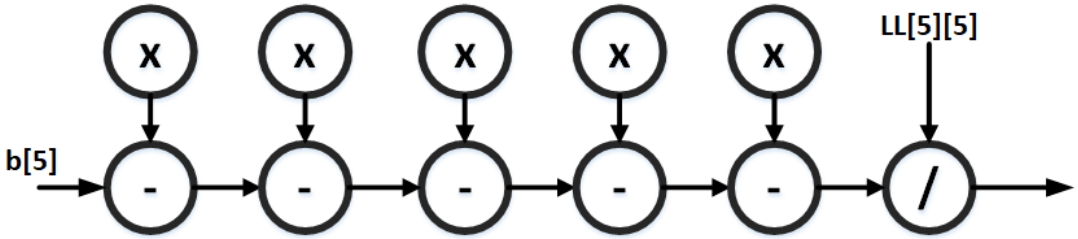


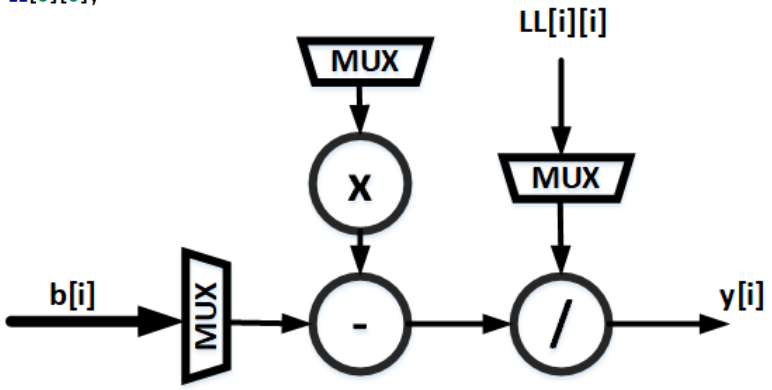
Resource Optimization(1/2)

- Reuse the hardware resource

```
for(int i=0;i<=5;i=i+1)
{
    T up = b[i];
    T down = LL[i][i];
    for(int j=0;j<i;j++)
    {
        up = up - y[j]*LL[i][j]
    }
    y[i] = up / down;
}
```

```
y[0] = (b[0]) / LL[0][0];
y[1] = (b[1] - LL[1][0] * y[0]) / LL[1][1];
y[2] = (b[2] - LL[2][0] * y[0] - LL[2][1] * y[1]) / LL[2][2];
y[3] = (b[3] - LL[3][0] * y[0] - LL[3][1] * y[1] - LL[3][2] * y[2]) / LL[3][3];
y[4] = (b[4] - LL[4][0] * y[0] - LL[4][1] * y[1] - LL[4][2] * y[2] - LL[4][3] * y[3]) / LL[4][4];
y[5] = (b[5] - LL[5][0] * y[0] - LL[5][1] * y[1] - LL[5][2] * y[2] - LL[5][3] * y[3] - LL[5][4] * y[4]) / LL[5][5];
```





Resource Optimization(2/2)

- Original: 15SUB+15MUL+6DIV (Also, timing vio)
- Optimized: 1SUB+1MUL+1DIV

FF: x 0.20 reduced

LUT: x 0.24 reduced

Cycles: x 1.8 increased (Very minor to total cycle, < 10% total cycle)

Performance Profile | Resource Profile

Name	BRAM	DSP	FF	LUT	Bits P0	B
backSubstitutionHLS_float_6_s	0	0	1373	1159		
I/O Ports(8)					256	
Instances(0)	0	0	0	0		
Memories(0)	0		0	0	0	
Expressions(0)	0	0	0	0	0	0
Registers(40)			1373		1373	
Channels(0)	0		0	0	0	
Multiplexers(19)	0		0	1159	522	
DSP(0)		0				

Performance Profile | Resource Profile

Name	BRAM	DSP	FF	LUT	Bits P0
backSubstitutionHLS_float_6_s	0	0	269	277	
I/O Ports(3)					96
Instances(2)	0	0	235	214	
Memories(0)	0		0	0	0
Expressions(6)	0	0	0	12	6
Registers(2)			2		2
Channels(1)	0		32	33	32
pingpong	0		32	33	32
y_U	0		32	33	32
Multiplexers(2)	0		0	18	2
DSP(2)		0			

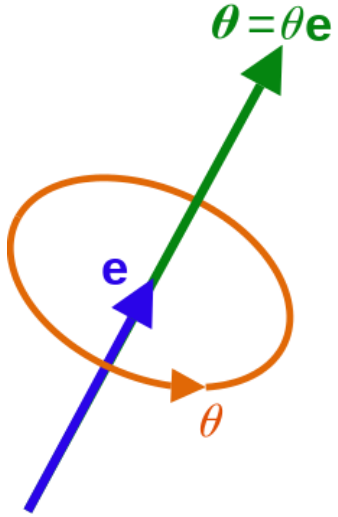
5. meanSquaredErrorHLS

Similar to CalJacobianMSEHLS

```
//// fetch SIMD points at once (However, memory is the bottleneck for performing SIMD parallel)
for (k = 0; k < N; k++) {

    //read data from stream
    //inPoint3D_x = point3D_x.read();
    //inPoint3D_y = point3D_y.read();
    //inPoint3D_z = point3D_z.read();
    //inMarkerPixelColor = marker_pixel_color.read();
    inPoint3D_x = point3D_x_buffer[k];
    inPoint3D_y = point3D_y_buffer[k];
    inPoint3D_z = point3D_z_buffer[k];
    inMarkerPixelColor = marker_pixel_color_buffer[k];

    //compute camera coordinate
    camCoord_x = (poseMat[0][0]) * inPoint3D_x + (poseMat[0][1]) * inPoint3D_y + (poseMat[0][2]) * inPoint3D_z
    camCoord_y = (poseMat[1][0]) * inPoint3D_x + (poseMat[1][1]) * inPoint3D_y + (poseMat[1][2]) * inPoint3D_z
    camCoord_z = (poseMat[2][0]) * inPoint3D_x + (poseMat[2][1]) * inPoint3D_y + (poseMat[2][2]) * inPoint3D_z

    //compute pixel coordinate
    camCoord_z_inv = 1. / camCoord_z;
    camCoord_x_div_z = camCoord_x * camCoord_z_inv;
    camCoord_y_div_z = camCoord_y * camCoord_z_inv;


    //pixelCoord_x = int((cameraParam[0] * camCoord_x_div_z + cameraParam[2] - anchor_x)[0]);
    //pixelCoord_y = int((cameraParam[1] * camCoord_y_div_z + cameraParam[3] - anchor_y)[0]);
    pixelCoord_x = int(cameraParam[0] * camCoord_x_div_z + cameraParam[2] - anchor_x);
    pixelCoord_y = int(cameraParam[1] * camCoord_y_div_z + cameraParam[3] - anchor_y);
```

6. computeRrodriguesHLS (1/3)

$$R = \exp(\theta\mathbf{K}) = \sum_{k=0}^{\infty} \frac{(\theta\mathbf{K})^k}{k!} = I + \theta\mathbf{K} + \frac{1}{2!}(\theta\mathbf{K})^2 + \frac{1}{3!}(\theta\mathbf{K})^3 + \cdots$$



that is,

$$R = I + (\sin\theta)\mathbf{K} + (1 - \cos\theta)\mathbf{K}^2 \ ,$$

```
T rrt[9] = { rx * rx, rxry, rxrz,
             rxry, ry * ry, ryrz,
             rxrz, ryrz, rz * rz };
T r_x[9] = { 0, -rz, ry,
             rz, 0, -rx,
             -ry, rx, 0 };

for (int i = 0; i < 9; i++) {
    R[i] = c * I[i] + c1 * rrt[i] + s * r_x[i];
}
```

6. computeRrodriguesHLS (2/3)

orignal host code:

Unroll the full 9x3 Rrodrigue Jacobian

```
double rxcl1_l4 = rx * cl1_l4;
double rycl1_l4 = ry * cl1_l4;
double rzcl1_l4 = rz * cl1_l4;
(*JRr)(0, 0) = -(sl * ry2rz2 * rx_l3) - (2 * ry2rz2 * rxcl1_l4);
(*JRr)(0, 1) = (2 * cl1 * ry_l2) - (sl * ry2rz2 * ry_l3)
    - (2 * ry2rz2 * rycl1_l4);
(*JRr)(0, 2) = (2 * cl1 * rz_l2) - (sl * ry2rz2 * rz_l3)
    - (2 * ry2rz2 * rzcl1_l4);
(*JRr)(1, 0) = (rzsl * rx_l3) - (rzcl * rx_l2) - (cl1 * ry_l2)
    + (rx * rysl * rx_l3) + (2 * rx * ry * rxcl1_l4);
(*JRr)(1, 1) = (rzsl * ry_l3) - (rzcl * ry_l2) - (cl1 * rx_l2)
    + (rx * rysl * ry_l3) + (2 * rx * ry * rycl1_l4);
(*JRr)(1, 2) = (rzsl * rz_l3) - (rzcl * rz_l2) - sl_l
    + (rx * rysl * rz_l3) + (2 * rx * ry * rzcl1_l4);
(*JRr)(2, 0) = (rycl * rx_l2) - (cl1 * rz_l2) - (rysl * rx_l3)
    + (rx * rzsl * rx_l3) + (2 * rx * rz * rxcl1_l4);
(*JRr)(2, 1) = sl_l + (rycl * ry_l2) - (rysl * ry_l3)
    + (rx * rzsl * ry_l3) + (2 * rx * rz * rycl1_l4);
(*JRr)(2, 2) = (rycl * rz_l2) - (cl1 * rx_l2) - (rysl * rz_l3)
    + (rx * rzsl * rz_l3) + (2 * rx * rz * rzcl1_l4);
(*JRr)(3, 0) = (rzcl * rx_l2) - (cl1 * ry_l2) - (rzsl * rx_l3)
    + (rx * rysl * rx_l3) + (2 * rx * ry * rxcl1_l4);
(*JRr)(3, 1) = (rzcl * ry_l2) - (cl1 * rx_l2) - (rzsl * ry_l3)
    + (rx * rysl * ry_l3) + (2 * rx * ry * rycl1_l4);
```

Code found in OpenCV github:

Multipliers, dividors are greatly reduced

Rolled in 27 cycles.

```
for (int i = 0; i < 9; i++) {
    R[i] = c * I[i] + c1 * rrt[i] + s * r_x[i];
}

T drrt[] = { rx + rx, ry, rz, ry, 0, 0, rz, 0, 0,
                      0, rx, 0, rx, ry + ry, rz, 0, rz, 0,
                      0, 0, rx, 0, 0, ry, rx, ry, rz + rz };
T d_r_x_[] = { 0, 0, 0, 0, 0, -1, 0, 1, 0,
                      0, 0, 1, 0, 0, 0, -1, 0, 0,
                      0, -1, 0, 1, 0, 0, 0, 0, 0 };
for (int i = 0; i < 3; i++)
{
    T ri = i == 0 ? rx : i == 1 ? ry : rz;
    T a0 = -s * ri, a1 = (s - 2 * c1_itheta) * ri, a2 = c1_itheta;
    T a3 = (c - s_itheta) * ri, a4 = s_itheta;
    for (int k = 0; k < 9; k++)
        JRr[i * 9 + k] = a0 * I[k] + a1 * rrt[k] + a2 * drrt[i * 9 + k] +
        a3 * r_x[k] + a4 * d_r_x_[i * 9 + k];
}
```

Modules & Loops	Issue Type	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT
denseAlignment_api		-0.22	-	-	-	-	-	no	122	493	105363	85679
denseAlignmentHLS_float_1_300_400_s	II Violation	-0.22	-	-	-	-	-	no	10	492	104421	83660
meanSquaredErrorHLS_float_1_300_400_s		-0.22	-	-	-	-	-	no	5	193	36127	33835
computegRrodriguesHLS_float_s		-0.22	1317	4.390E3	-	1317	-	no	0	86	12246	16720
sin_or_cos_double_s		0.00	53	177.000	-	53	-	no	0	43	5435	7211
VITIS_LOOP_802_1		-	27	89.991	1	-	27	no	-	-	-	-
VITIS_LOOP_805_2		-	9	29.997	1	-	9	no	-	-	-	-
VITIS_LOOP_855_3		-	189	630.000	21	-	9	no	-	-	-	-
VITIS_LOOP_865_4		-	963	3.210E3	321	-	3	no	-	-	-	-
VITIS_LOOP_870_5		-	315	1.050E3	35	-	9	no	-	-	-	-

6. computeRrodriguesHLS (3/3)

orignal host code:

Unroll the full 9x3 Rrodrigue Jacobian

Summary

Name	BRAM_18K	DSP	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	182	-
FIFO	-	-	-	-	-
Instance	0	129	14701	16708	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	2560	-
Register	-	-	3079	-	-
Total	0	129	17780	19450	0
Available	4320	6840	2364480	1182240	960
Available SLR	1440	2280	788160	394080	320
Utilization (%)	0	1	~0	1	0
Utilization SLR (%)	0	5	2	4	0

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
75	176	0.250 us	0.587 us	75	176	none

Cycles: 176, DSP:129, FF: 17780, LUT:19450

Code found in OpenCV github:

Rolled in 27 cycles.

Summary

Name	BRAM_18K	DSP	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	363	-
FIFO	-	-	-	-	-
Instance	0	86	10870	14422	-
Memory	0	-	160	165	-
Multiplexer	-	-	-	1770	-
Register	-	-	1216	-	-
Total	0	86	12246	16720	0
Available	4320	6840	2364480	1182240	960
Available SLR	1440	2280	788160	394080	320
Utilization (%)	0	1	~0	1	0
Utilization SLR (%)	0	3	1	4	0

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
75	1317	0.250 us	4.390 us	75	1317	none

Cycles: 1317, DSP:86, FF: 12246, LUT:16720

DSP: x 0.66 reduced

FF: x 0.68 reduced

LUT: x 0.86 reduced

Cycles: x 7.4 increased

(Due to additional temporal storages)

(Very minor to total cycle, < 10% total cycle)

Overview

- Direct Method Pose Optimization Problem
- Original Host Code
- Objective: Target to achieve
- System Diagram
- Kernel Code + Optimization
- Evaluation
- Application Demo
- Summary
- Future Work
- Appendix

Evaluation (Break down)

Original

Timing Vio

Timing Vio

Kernel	calJacobianMse	calHessianJres	cholesky	backSubstitution	Rodrigues
Cycles	120K	60.1K	0.9K	**0.5K**	0.2K
Latency(ns)	401K	282K	3.1K	**2.4K**	0.6K

Partial Optimized

Kernel	calJacobianMse	calHessianJres	cholesky	backSubstitution	Rodrigues
Cycles	**10.3K**	**10.6K**		0.9K	1.3K
Latency(ns)	**34K**	**35K**		3.1K	4.4K

Demo System

(ms)	CPU	U50
Kernel	94.4	13.57
Memory Write		0.060
Memory Read		0.077

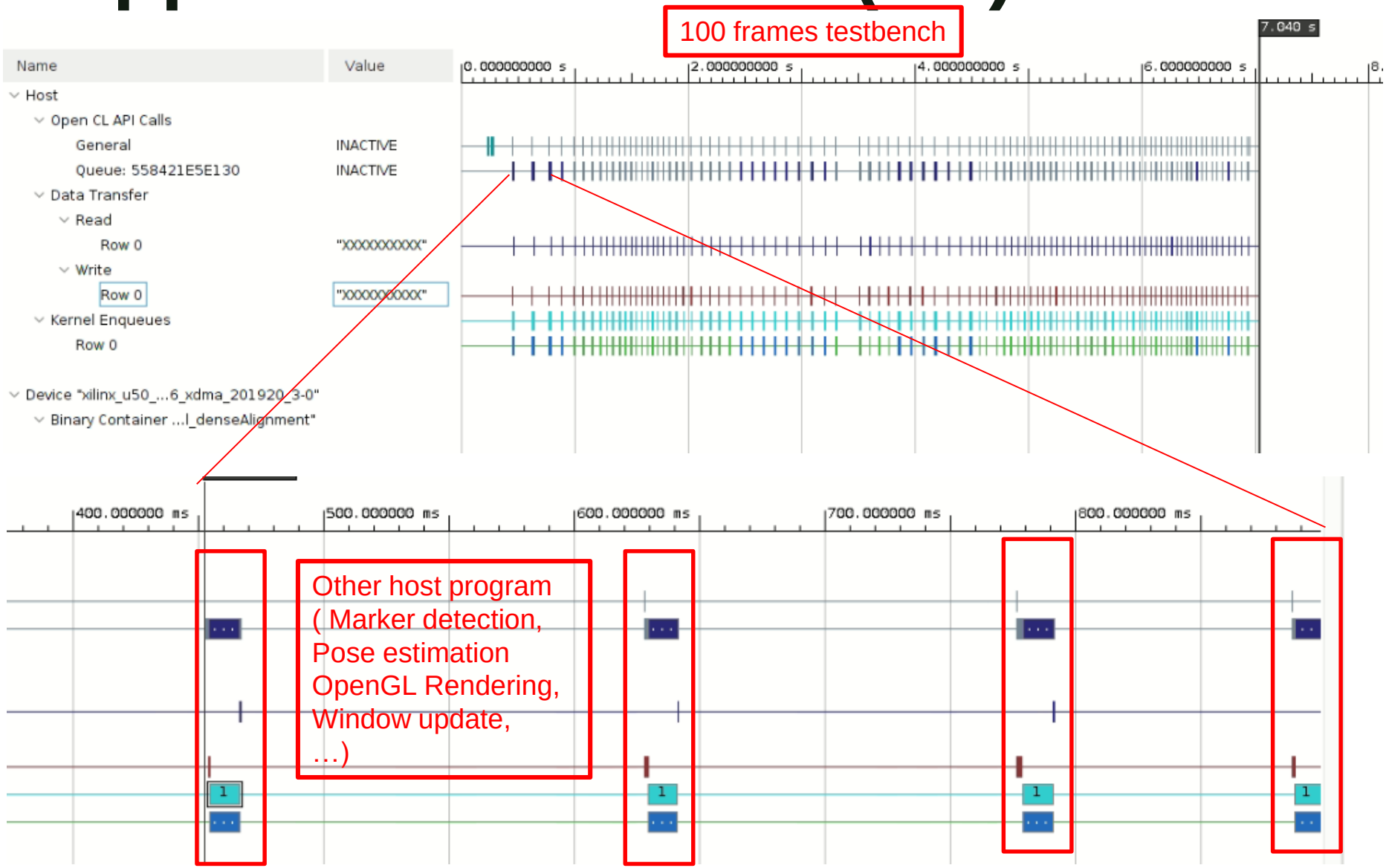
FPS: 73.7 **x 7.0 Improved**

Data transfer / Kernel ratio: 0.57% (Acceptable !)

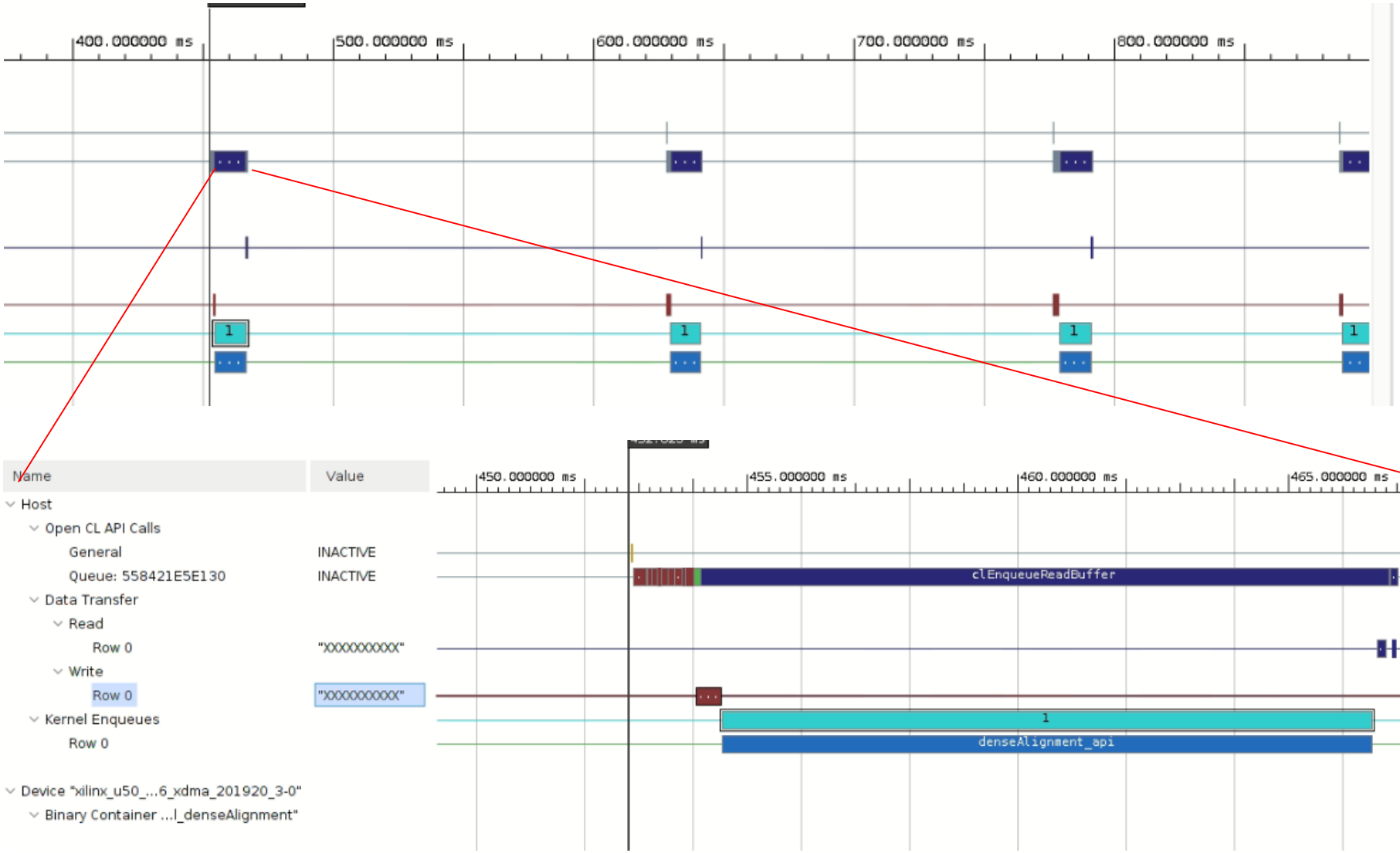
Host Transfer

Context:Number of Devices	Transfer Type	Number of Buffer Transfers	Transfer Rate (MB/s)	Avg Bandwidth Utilization (%)	Avg Size (KB)	Total Time (ms)	Avg Time (ms)
context0:1	READ	182	0.390	0.004	0.030	14.017	0.077
context0:1	WRITE	1081	757.752	7.893	45.618	65.078	0.060

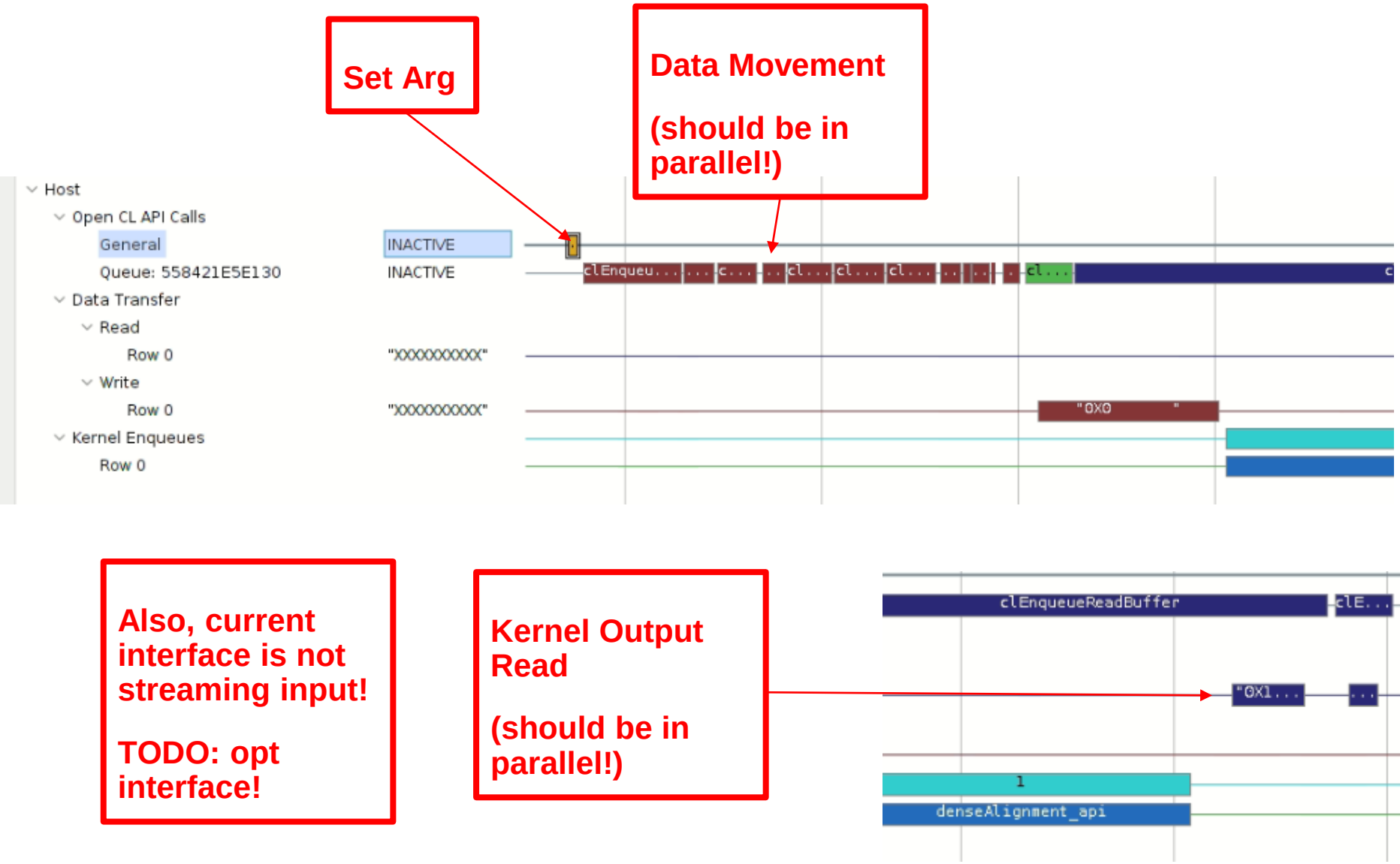
Application Timeline (1/3)



Application Timeline (2/3)



Application Timeline (3/3)



About Timing Violation

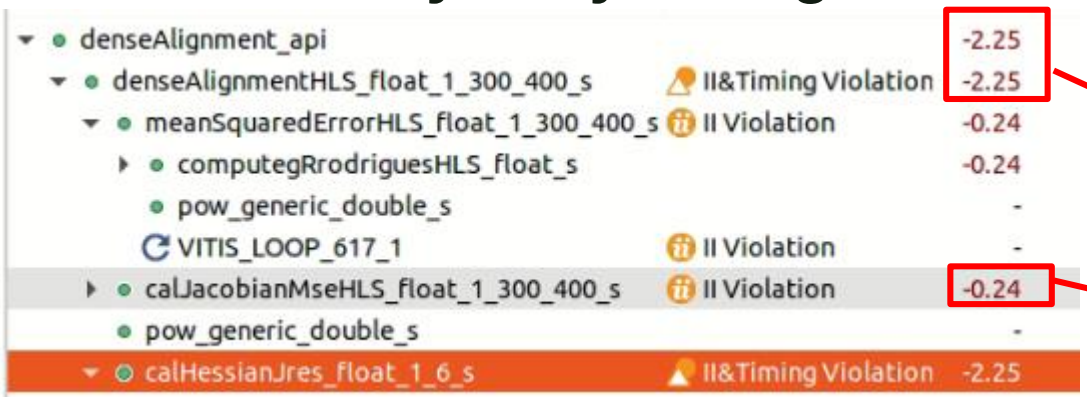
- If there is timing violation, then the answers might be wrong

- **Vitis may/may not give warnings**

  ***ERROR* Current xclbin is in-use, can't change**

  ***invalid %N$ use detected**

- **It is usually okay if negative slack < 10% period**



This is not okay!

Clock: 3.3ns (300MHz)

This is okay!

Overview

- Direct Method Pose Optimization Problem
- Original Host Code
- Objective: Target to achieve
- System Diagram
- Kernel Code + Optimization
- Evaluation
- Application Demo
- Summary
- Future Work
- Appendix

Application Demo

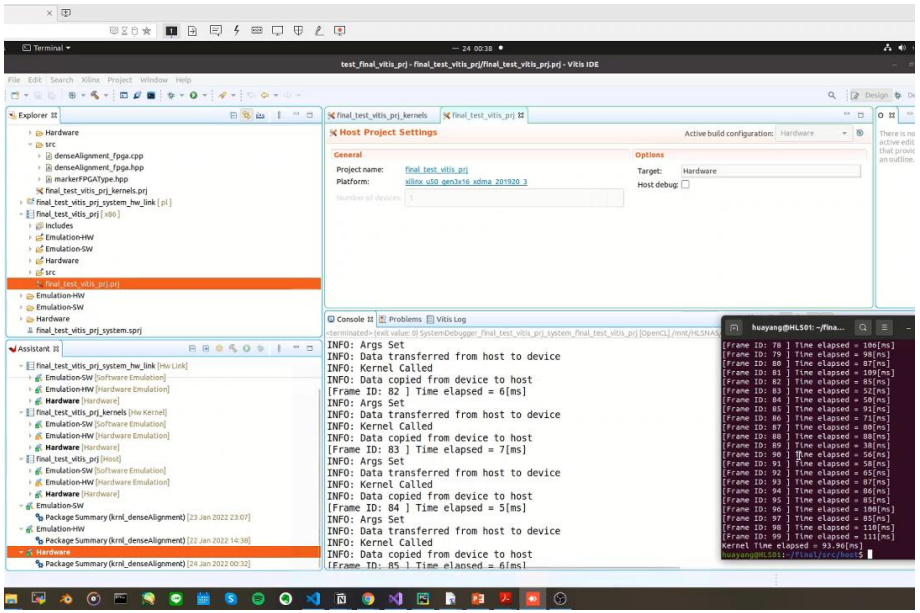
Pose Accuracy:

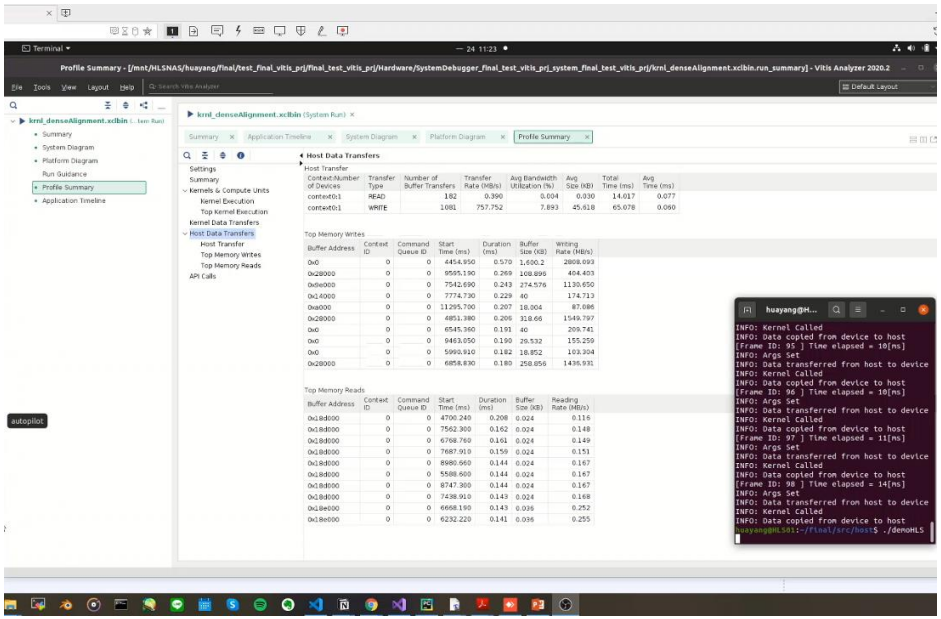
Before DPR: Rotation: **0.44 degree** Translation: **5.88 mm**

After DPR : Rotation: **0.38 degree** Translation: **1.06 mm**

Original Time: 94.4ms

Hardware Accelerated Time: 13.57ms





Summary

- Given an original C++ code (highly software oriented)
  - Find the bottleneck
  - Rewrite to synthesizable code
  - Optimize it with kernel code with II=1
- FPS: 74 fps @ 300MHz (N = 10000, M = 6)
- Implemented in OpenCL & U50 system
  - Demo system with many other non-synthesizable host code libraries
- SW emulation, hardware passed testbench
- A baseline model for future optimization

Future Work

- Fixed point method
- Add bilinear interpolation
- Host code optimization (non-blocking)
- Eliminate on-chip memory
- Host data compression & kernel decompression
- fixed point method
- Unroll @ the N dimension
- Extension to C channel frame/gradient/template input
- Extension to scalable M
  - Currently, unroll M, however if M >> 6, not feasible

Overview

- Direct Method Pose Optimization Problem
- Original Host Code
- Objective: Target to achieve
- System Diagram
- Kernel Code + Optimization
- Evaluation
- Application Demo
- Summary
- Future Work
- Appendix

Appendix

- Project Flow
- Problems with Other Libraries

Project Flow

- Host code re-write (no opencl)
  - Change non-synthesizable code (e.g. Eigen)
  - Manually include HLS libraries (e.g. hls_vector, hls_stream)
  - Compile with g++ (include all non-synthesizable libraries, e.g. opengl)
  - End-to-end debug with c++ testbench
- Add OpenCL and perform software emulation in Vitis or g++
- End-to-end analysis
  - Change to optimized kernel code
  - Hardware build
  - Analyze with Vitis & Vitis HLS
- Kernel code synthesis & optimization
  - Optimize each kernel in Vitis_hls

Demo Environment & Host code

Partially contain kernel code

```
#ifndef __DENSE_ALIGNMENT_FPGA__
#define __DENSE_ALIGNMENT_FPGA__

// #include "ap_int.h"
// #include "hls_stream.h"
// #include "assert.h"


#include "hls_stream.h"

//#include <hls_math.h>
#include <cmath>
#include "markerFPGAType.hpp"
```

main.cpp

```
Edit Selection View Go Run Terminal Help
stricted Mode is intended for safe code browsing. Trust this window to enable all features. Mana
denseAlignment_fpga.hpp × calError.py demoHLS.cpp ×
mnt > HLSNAS > huayang > final > src > host > demoHLS.cpp

#include <glad/glad.h>
#include <GLFW/glfw3.h>
#include <opencv2/opencv.hpp>
#include <Eigen/Core>
#include <opencv2/core/eigen.hpp>
#include <opencv2/aruco.hpp>

#include <window.h>
#include <rendererEngine.h>
#include <camera.h>
#include <shader.h>
#include <model.h>
#include <light.h>
#include <transformation.h>
//#include <dodecaSystemTracker.h>
#include <dodecaSystemTrackerHLS.h>
#include <iostream>
#include <array>
#include <boost/log/trivial.hpp>


using std::chrono::duration_cast;
using std::chrono::milliseconds;
using std::chrono::seconds;
using std::chrono::system_clock;

const unsigned int SCR_WIDTH = 1280;
const unsigned int SCR_HEIGHT = 1024;
```

```
// 1. create window
Window window1 = Window(SCR_WIDTH, SCR_HEIGHT, "demo window");


// 2. create shaders
Shader modelShader(vModelShaderPath.c_str(), fModelShaderPath.c_str());
Shader lightCubeShader(vLightCubeShaderPath.c_str(), fLightCubeShaderPath.c_str());

// 3. create lights
std::vector<Light*> lights;
Eigen::Vector3f ambient = Eigen::Vector3f(0.1f, 0.1f, 0.1f);
lights.push_back(new Light(Eigen::Vector3f(-100.0f, 0.0f, 0.0f), ambient));
lights.push_back(new Light(Eigen::Vector3f(100.0f, 0.0f, 0.0f), ambient));
lights.push_back(new Light(Eigen::Vector3f(0.0f, -100.0f, 0.0f), ambient));
lights.push_back(new Light(Eigen::Vector3f(0.0f, 100.0f, 0.0f), ambient));
lights.push_back(new Light(Eigen::Vector3f(0.0f, 0.0f, -100.0f), ambient));
lights.push_back(new Light(Eigen::Vector3f(0.0f, 0.0f, 400.0f), ambient));


// 4. create models
std::vector<Model*> models;
//models.push_back(new Model("dodeca", "D:\\MultimediaIClab\\AR\\BrainSurgery\\Checkp
models.push_back(new Model("dodeca1", dodecaModelPath1, 1.f));
models.push_back(new Model("dodeca2", dodecaModelPath2, 1.f));
//models.push_back(new Model("backpack1", "data/backpack/backpack.obj", 10.f));

// 5. create camera
//(globally created here)

// 6. create renderer engine
RendererEngine rendererEngine1 = RendererEngine(&window1, models, lights, &cam1, &moc
```

Demo Environment & Host code

Before rendering loop

```
// create a kernel:
OCL_CHECK(err, cl::Kernel kernel(program, "denseAlignment_api", &err));
std::cout << "INFO: Kernel Created" << std::endl;

// Allocate the buffers:
OCL_CHECK(err, cl::Buffer buffer_point3D_x(context, CL_MEM_READ_ONLY, p
OCL_CHECK(err, cl::Buffer buffer_point3D_y(context, CL_MEM_READ_ONLY, p
OCL_CHECK(err, cl::Buffer buffer_point3D_z(context, CL_MEM_READ_ONLY, p
OCL_CHECK(err, cl::Buffer buffer_marker_pixel_color(context, CL_MEM_REA

OCL_CHECK(err, cl::Buffer buffer_frame(context, CL_MEM_READ_ONLY, frame_
OCL_CHECK(err, cl::Buffer buffer_gradient_gx(context, CL_MEM_READ_ONLY,
OCL_CHECK(err, cl::Buffer buffer_gradient_gy(context, CL_MEM_READ_ONLY,

OCL_CHECK(err, cl::Buffer buffer_cameraParam(context, CL_MEM_READ_ONLY,
OCL_CHECK(err, cl::Buffer buffer_validRegion(context, CL_MEM_READ_ONLY,
OCL_CHECK(err, cl::Buffer buffer_JRr(context, CL_MEM_READ_ONLY, JRr_size
OCL_CHECK(err, cl::Buffer buffer_p(context, CL_MEM_READ_WRITE , p_size_b
OCL_CHECK(err, cl::Buffer buffer_R(context, CL_MEM_READ_WRITE , R_size_b
std::cout << "INFO: Buffers Created" << std::endl;


// Set kernel arguments:
OCL_CHECK(err, err = kernel.setArg(0, buffer_point3D_x));
OCL_CHECK(err, err = kernel.setArg(1, buffer_point3D_y));
OCL_CHECK(err, err = kernel.setArg(2, buffer_point3D_z));
OCL_CHECK(err, err = kernel.setArg(3, buffer_marker_pixel_color));

OCL_CHECK(err, err = kernel.setArg(4, buffer_frame));
OCL_CHECK(err, err = kernel.setArg(5, buffer_gradient_gx));
OCL_CHECK(err, err = kernel.setArg(6, buffer_gradient_gy));
```

Kernel call in Rendering Loop

```
OCL_CHECK(err, mQueue.enqueueWriteBuffer(mBuffer_point3D_x, CL_TRUE, 0, num*sizeof(dataType)
OCL_CHECK(err, mQueue.enqueueWriteBuffer(mBuffer_point3D_y, CL_TRUE, 0, num*sizeof(dataType)
OCL_CHECK(err, mQueue.enqueueWriteBuffer(mBuffer_point3D_z, CL_TRUE, 0, num*sizeof(dataType)
OCL_CHECK(err, mQueue.enqueueWriteBuffer(mBuffer_marker_pixel_color, CL_TRUE, 0, num*sizeof(

OCL_CHECK(err, mQueue.enqueueWriteBuffer(mBuffer_frame, CL_TRUE, 0, h*w*sizeof(dataType), I_
OCL_CHECK(err, mQueue.enqueueWriteBuffer(mBuffer_gradient_gx, CL_TRUE, 0, h*w*sizeof(dataTyp
OCL_CHECK(err, mQueue.enqueueWriteBuffer(mBuffer_gradient_gy, CL_TRUE, 0, h*w*sizeof(dataTyp

OCL_CHECK(err, mQueue.enqueueWriteBuffer(mBuffer_cameraParam, CL_TRUE, 0, mCameraParam_size_
OCL_CHECK(err, mQueue.enqueueWriteBuffer(mBuffer_validRegion, CL_TRUE, 0, mValidRegion_size_
OCL_CHECK(err, mQueue.enqueueWriteBuffer(mBuffer_JRr, CL_TRUE, 0, mJRr_size_bytes, &JRr_hls[
OCL_CHECK(err, mQueue.enqueueWriteBuffer(mBuffer_p, CL_TRUE, 0, mP_size_bytes, &p_hls[0], nu
OCL_CHECK(err, mQueue.enqueueWriteBuffer(mBuffer_R, CL_TRUE, 0, mR_size_bytes, &R_hls[0], nu
std::cout << "INFO: Data transferred from host to device" << std::endl;

// Execute the kernel:
OCL_CHECK(err, err = mQueue.enqueueTask(mKernel));
std::cout << "INFO: Kernel Called" << std::endl;

// Copy Result from Device Global Memory to Host Local Memory
// queue.enqueueReadBuffer(buffer_outImage, // This buffers data will be read
//                         CL_TRUE,         // blocking call
//                         0,               // offset
//                         image_out_size_bytes,
//                         image_output.data, // Data will be stored here
//                         nullptr, &event);
mQueue.enqueueReadBuffer(mBuffer_p, CL_TRUE, 0, mP_size_bytes, &p_hls[0], nullptr, &event);
mQueue.enqueueReadBuffer(mBuffer_R, CL_TRUE, 0, mR_size_bytes, &R_hls[0], nullptr, &event);
std::cout << "INFO: Data copied from device to host" << std::endl;
```

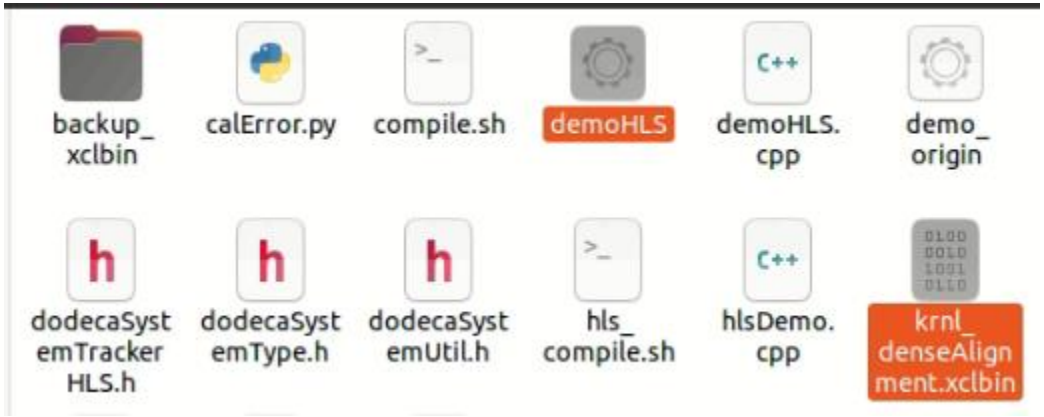
Main Rendering Loop

```
// render loop
// -----------
 //while (!glfwWindowShouldClose(window1.mglfwWindow))
for (int i = 0; i < NUM; i++)
// for (int i = 0; i < 61; i++)
{
    // read image
    //cv::Mat im = cv::imread(imPath + "\\" +  ZeroPadNumber(i,2) + ".png");
    //cv::Mat im_gray = cv::imread(imPath + "\\" + ZeroPadNumber(i, 2) + ".png", cv::IMREAD_GRAYSCALE);
    cv::Mat im = cv::imread(imPath + "/" + ZeroPadNumber(i, 2) + ".png");
    cv::Mat im_gray = cv::imread(imPath + "/" + ZeroPadNumber(i, 2) + ".png", cv::IMREAD_GRAYSCALE);
```

Demo Environment & Host code

Compile with g++

```
g++ -g -Wall -DBOOST_LOG_DYN_LINK
./demoHLS.cpp
./xcl2.cpp
./dodecaSystemTrackerHLS.cpp
../kernel/denseAlignment_fpga.cpp
./RenderEngine/utils.cpp
./RenderEngine/rendererEngine.cpp
./RenderEngine/camera.cpp
./RenderEngine/transformation.cpp
./RenderEngine/shader.cpp
/mnt/HLSNAS/huayang/vclib/glad/src/glad.c
-I/mnt/HLSNAS/huayang/labC/lab_c/libs/xf_opencv/ext/xcl2
-I /mnt/HLSNAS/huayang/vclib/eigen-3.4.0
-I /mnt/HLSNAS/huayang/vclib/glfw-3.3.6/include
-I /mnt/HLSNAS/huayang/vclib/glad/include
-I /mnt/HLSNAS/huayang/vclib/opencv/build/include/opencv4
-I /mnt/HLSNAS/huayang/vclib/glm-0.9.9.7/glm
-I /mnt/HLSNAS/huayang/final/src/host/RenderEngine
-I /mnt/HLSNAS/huayang/final/src/host
-I /mnt/HLSNAS/huayang/final/src/kernel
-I /mnt/HLSNAS/huayang/vclib/assimp-5.0.1/include
-I /mnt/HLSNAS/huayang/vclib/boost_1_78_0
-I/opt/Xilinx/Vivado/2020.2/include/
-I/opt/xilinx/xrt/include/
-lboost_log -lboost_thread  -lboost_filesystem -lpthread
/mnt/HLSNAS/huayang/vclib/glfw-3.3.6/build/src/libglfw3.a
-ldl -lX11
-L /mnt/HLSNAS/huayang/vclib/assimp-5.0.1/lib -lassimp
-L /mnt/HLSNAS/huayang/vclib/opencv/install/lib
-L /opt/xilinx/xrt/lib
-lopencv_core -lopencv_imgcodecs -lopencv_highgui -lopencv_imgproc -lopencv_calib3d
-lopencv_aruco -lopencv_video -lxilinxopencl -lpthread -lrt -lstdc++ -o demoHLS
```




```
huayang@HLS01:~/final/src/host$ ./demoHLS
```

```
// Load binary:
std::string binaryFile = xcl::find_binary_file(device_name, "krnl_denseAlignment");
cl::Program::Binaries bins = xcl::import_binary_file(binaryFile);
devices.resize(1);
OCL_CHECK(err, cl::Program program(context, devices, bins, NULL, &err));
std::cout << "INFO: Program Created" << std::endl;
```

Problems with Other Libraries

- If one uses Boost Library, be cautious to the version
- Vitis uses its own Boost library (certain version)

- If the latest version is used, OpenCL might not found the FPGA board

ERROR: double free or corruption (out)

  - After gdb step in debug, it stops at opencl library.
  - error at wrong Boost::FileSystem version

```
xcl::get_devices (vendor_name="Xilinx") at ./xcl2.cpp:38
38          cl::Platform::get(&platforms);
(gdb) s
cl::Platform::get (
    platforms=0x555555594796 <std::_Vector_base<cl::Platform, s
:Platform> >::_Vector_impl::_Vector_impl()+40>) at /usr/include
2592        static cl_int get(
(gdb) n
2595            cl_uint n = 0;
(gdb) n
2597            if( platforms == NULL ) {
(gdb) n
2601            cl_int err = ::clGetPlatformIDs(0, NULL, &n);
(gdb)
```