Eikonal Parallel Solver on Android

- Eikonal equation
- C++ Implementation on serial and multi threaded CPUs
- Code optimization aspects and results
- G++,clang++ and Intel compiler
- C++ parallel programming on Android using NDK
- Android Programming

Eikonal equation

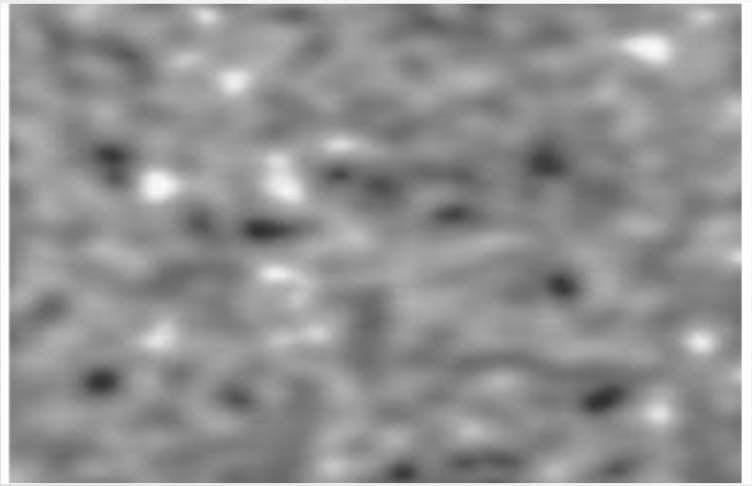
- Mathematical description.
- Definition of the local solver.
- Algorithm description

Mathematical description

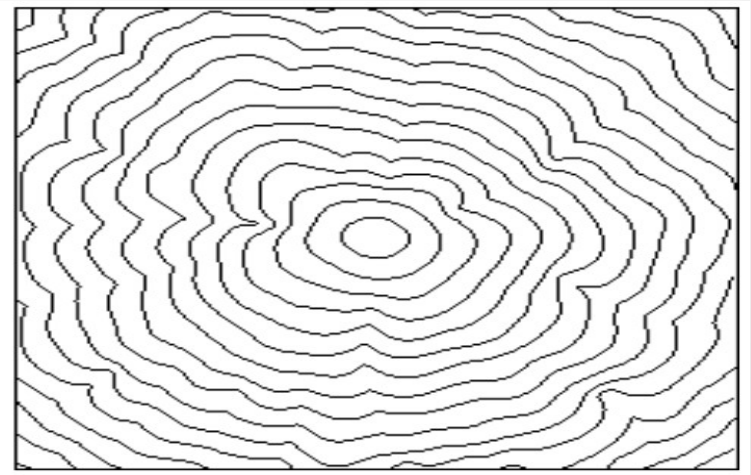
- Special case of Hamilton-Jacobi PDEs.
- Compute time of excitation $\varphi(x) : R^3 \rightarrow R$ for every point $x \in \Omega$ of the mesh discretized by tetrahedral elements

$$H(x, \nabla\varphi) = \sqrt{(\nabla\varphi)^T M(\nabla\varphi)} = 1 \ \forall x \in \Omega \subset R\ 3$$

with with M(x) denoting the velocity information of the domain (a.k.a velocity map).



Velocity map example



Solution example

Definition of the local solver.

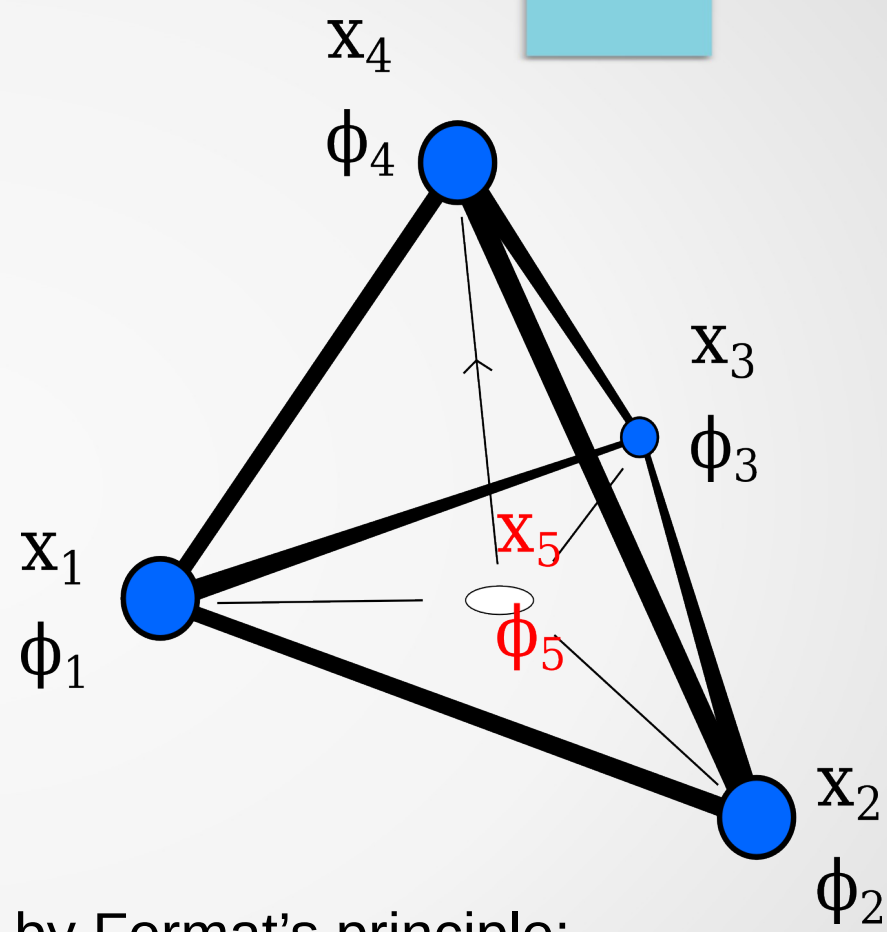
$\Phi_{i,j} = \Phi_j - \Phi_i$ Travel time

$e_{i,j} = x_j - x_i$ Edge vector

$X_i = (x,y,z)^T$ where $x,y,z \in R$

Upwind scheme to compute Φ_4.

Speed function is constant so the travel time to x_4 is determined by the time associated with the segment along the wave front normal direction that minimizes Φ_4.



Denote the travel time from x_5 to x_4 given by Fermat's principle:

$$\Phi_{5,4} = \Phi_4 - \Phi_5 = \sqrt{e_{5,4}^T M e_{5,4}}$$

Definition of the local solver.

- **The goal** is to find the location of x_5 that minimizes Φ_4

$x_5 = \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3$

$\Phi_5 = \lambda_1 \Phi_1 + \lambda_2 \Phi_2 + \lambda_3 \Phi_3$

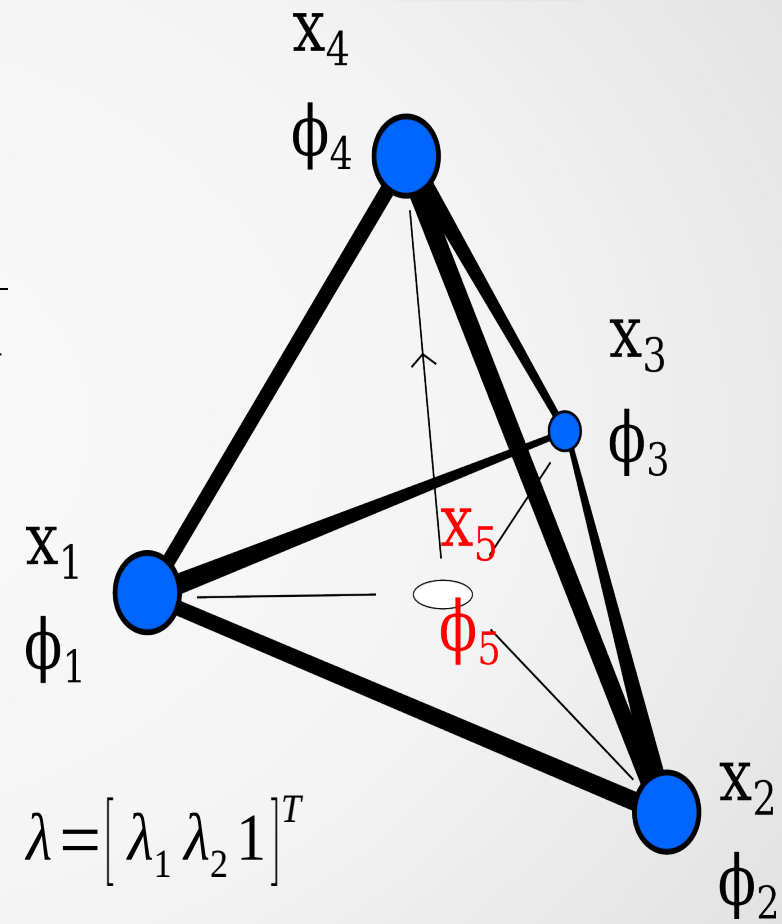
$1 = \lambda_1 + \lambda_2 + \lambda_3$

$\Phi_4 = \lambda_1 \Phi_1 + \lambda_2 \Phi_2 + (1 - \lambda_1 - \lambda_2)\Phi_3 + \sqrt{e_{5,4}^T M e_{5,4}}$

Take partial derivatives w.r.t λ_1, λ_2 and equate them to 0:

$$\begin{cases} \Phi_{1,3}\sqrt{\lambda^T M' \lambda} = \lambda^T \alpha \\ \Phi_{2,3}\lambda^T \alpha = \Phi_{1,3}\lambda^T \beta \end{cases}$$

$$M' = \begin{cases} \alpha = \left[e_{1,3}^T M e_{13}, e_{2,3}^T M e_{13}, e_{3,4}^T M e_{13} \right]^T \\ \beta = \left[e_{1,3}^T M e_{23}, e_{2,3}^T M e_{23}, e_{3,4}^T M e_{23} \right]^T \\ \theta = \left[e_{1,3}^T M e_{34}, e_{2,3}^T M e_{34}, e_{3,4}^T M e_{34} \right]^T \end{cases}$$



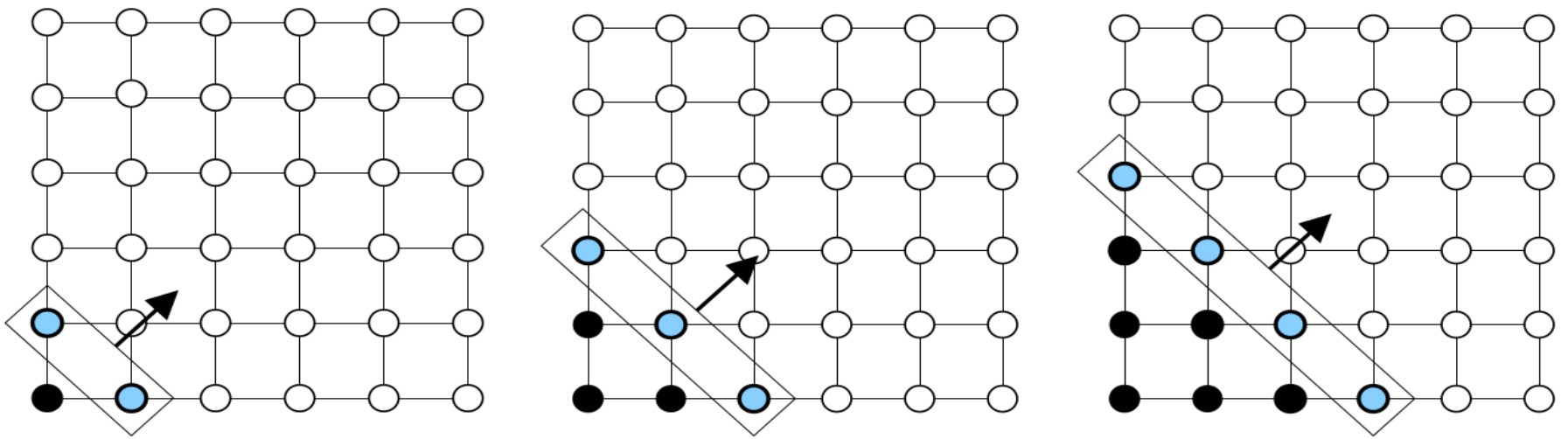
$$\lambda = \left[\lambda_1 \lambda_2 1 \right]^T$$

Apply the 2D local solver to the faces $\Delta_{124}\Delta_{134}\Delta_{234}$ in the following cases:

- No root.
- Bad tetrahedron
- $\lambda_i \notin [0, 1]$

Algorithm description – FIM method

- Iterate over elements in ”Active-List”
- Add not converged neighboring elements
- Remove converged elements



(a) Initial stage (b) After first update (c) After second update

Algorithm description – FIM method

- **Active List Update Scheme**
- Combined with the local solver for tetrahedral meshes
- Data structure: “Active list” set
- The algorithm is iterative
- During each iteration active vertices can be updated in parallel
- Convergence of the algorithm is proven on [1]

comment: 1. Initialization ($\mathbf{X}$: set of grid points, L : active list)

for each $\mathbf{x} \in \mathbf{X}$
do { **if** $\mathbf{x}$ is source
  then $U_{\mathbf{x}} \leftarrow 0$
  else $U_{\mathbf{x}} \leftarrow \infty$

for each $\mathbf{x} \in \mathbf{X}$
do { **if** any neighbor of $\mathbf{x}$ is source
  then add $\mathbf{x}$ to L

comment: 2. Update points in L

while L is not empty
do {
  for each $\mathbf{x} \in L$
  do {
    $p \leftarrow U_{\mathbf{x}}$
    $q \leftarrow g(U_{\mathbf{x}})$
    $U_{\mathbf{x}} \leftarrow q$
    if $|p - q| < \epsilon$
    then {
      for each 1-neighbor $\mathbf{x}_{nb}$ of $\mathbf{x}$
      do {
        if $\mathbf{x}_{nb}$ is not in L
        then {
          $p \leftarrow U_{\mathbf{x}_{nb}}$
          $q \leftarrow g(U_{\mathbf{x}_{nb}})$
          if $p > q$
          then {
            $U_{\mathbf{x}_{nb}} \leftarrow q$
            add $\mathbf{x}_{nb}$ to L
      remove $\mathbf{x}$ from L

Algorithm (1)

•Iterative C++ Implementation

- Update method
  - **Active list** initialized
  - **Set** used as the data structure for the active list.
  - List converted to vector for each iteration to avoid iterator invalidation during updates(insert & delete)
  - Iteratively **process** each node in the list
  - Return the solution vector **U**.
  - Generate .vtk file

```
vector<float> updateScheme::update(){

    initialize_list();

    while(!activeList.empty()){

        cout<<"ActiveList size: "<< activeList.size()<<endl;
        const vector<int> vec(activeList.cbegin(), activeList.cend());
        const size_t nvec=vec.size();

        for(size_t node=0;  node<nvec; ++node)
            process(vec[node]);
    }


    return getU();
}
```

•Iterative C++ Implementation

- Process method:
  - Calculate new solution and update the old value
  - Check convergence
  - Add neighboring elements
  - Avoid iterator invalidation:
    - Iteration done on the vector
    - Updates done on the set
  - Remove converged element

```
void updateScheme::process(const int & list_iter){

        float p= getU()[list_iter];
        float q= minSolutionOneRingTetra(list_iter); //calculate the solution of the point x
        getU()[list_iter] = q;

        if(abs(p-q)< epsilon*(1+(abs(p)+abs(q))/2)){ //if converges then

                for(int i=getN2nRow()[list_iter];i<getN2nRow()[list_iter+1];i++){ // for eac

                        if(getN2nCol()[i] != list_iter){

                                if(activeList.find(getN2nCol()[i]) == activeList.end()){

                                        p= getU()[getN2nCol()[i]];

                                        if(p > getU()[list_iter]) {// add only down wind nei

                                                q= minSolutionOneRingTetra(getN2nCol()[i]);

                                                if(p>q){ // p > getU()[*list_iter] add only
                                                        getU()[getN2nCol()[i]]=q;
                                                        activeList.insert(getN2nCol()[i]); /
                                                }
                                        }
                                }
                        }
                }
                activeList.erase(list_iter); //remove x from L
        }
}
```

OpenMP parallelization

- Because the serial algorithm does not depend significantly on the ordering of updates

  the extension to multiple processors is immediate.

- We simply divide the active list arbitrarily into N sublists, assign the sublists to the N threads,

  and let each thread use an asynchronous update for the vertices within the sublist.

- These updates are done by applying the updating step in Algorithm (1) to each sub-active list.

```
vector<float> updateScheme::update(){

    initialize_list();

    //the algorithm ends when all nodes are removed (converged)
    while(!activeList.empty()){

        int maxNrThread =8; // define the number of threads and also the number of the sublists

        int nr=activeList.size()/maxNrThread; //the number each sublist should contain.

        vector<set<int>> vectorSet; // every sublist is going to be contained in a vector
        vectorSet.resize(maxNrThread);

        vector<int> activeVector(activeList.cbegin(), activeList.cend());

        vector<int>::iterator it= activeVector.begin();

        for(int i=0;i<maxNrThread-1;i++){ // divide the big set equally for n-1 subsets.
            set<int> tmp(it,it+nr);
            vectorSet[i]=tmp;
            it=it+nr;
        }
        // The last one gets what it is left.
        set<int> tmp(it,activeVector.end());
        vectorSet[maxNrThread-1]=tmp;

        #pragma omp  parallel
        {

            #pragma omp for schedule(dynamic)
            for(int j=0;j<maxNrThread;j++){ // each thread is assigned a subset to process

                //convertion of the subset to a vector for a faster iteration oon the list
                //but still the udpate actions are done on the set for preventing iterator or references invalidation
                const vector<int> vec(vectorSet[j].cbegin(), vectorSet[j].cend());
                const size_t nvec=vec.size();

                for(size_t node=0;  node<nvec; ++node)
                    process(vec[node],vectorSet[j]);
            }
        }

        //clear the big list before merging all the updated sublists into one after the iteration.
        activeList.clear();
        for(int i=0;i<maxNrThread;i++){
            activeList.insert(vectorSet[i].cbegin(),vectorSet[i].cend());
        }

    }

    return getU();
}
```

OpenMP parallelization

```
void updateScheme::process(const int & list_iter,set<int>& activeList){

    float p,d,q;

      p= getU()[list_iter];
      q= minSolutionOneRingTetra(list_iter,p); //calculate the solution of the point x of the list by the local sover on every one nbh tetra and pick the min

      if(p>q){
          #pragma omp critical // this block should be critical because two threads may read the same value from the solution vector before one of them modifies it
                               //meaning that the condition p>q is as a result not valid since the comparison is done in non consistent data and therefore may happen that
                               // one thread updates teh solution even if it is not less than the current value
          {
                  if(q<getU()[list_iter])

                      #pragma omp atomic write
                      getU()[list_iter] = q;
          }
      }

      if(abs(p-q)< epsilon*(1+(abs(p)+abs(q))/2)){ //if converges then

          for(int i=getN2nRow()[list_iter];i<getN2nRow()[list_iter+1];i++){ // for each nbh if it is downwind than solve and remove the element.
              const int idx=getN2nCol()[i];

              if(idx != list_iter){

                  if(activeList.find(getN2nCol()[i]) == activeList.cend()){
```

OpenMP parallelization

```
/*
                    #pragma omp critical
                    {*/
                        p= getU()[idx];

                        d=getU()[list_iter];
                    //}

                    if(p > d) {// add only down wind neighbors

                        q= minSolutionOneRingTetra(idx,p); // calculate the solution of the "nbh of x" of the list by the local sover on every one nbh tetra and pick the min

                        if(p>q){ // if the new value computed is less than the current one for the nbh than the nbh should be added to the list.

                            #pragma omp critical
                            {

                                if(q<getU()[idx])

                                    #pragma omp atomic write
                                    getU()[idx]=q;

                            }

                            activeList.insert(idx); // insert the new element before the *list_iter elem
                        }
                    }


                }
            }
        }
        activeList.erase(list_iter); //remove x from L
    }
}
```

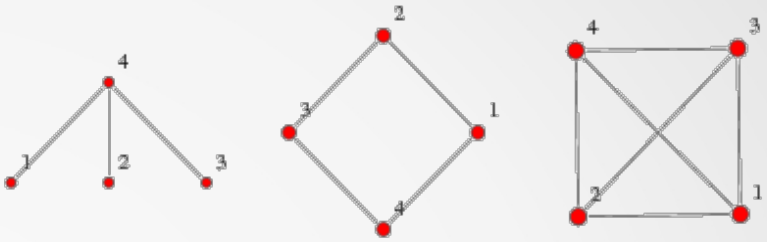
Code optimizations & Results

- Adjacency matrix method.
- #pragma omp critical optimization.
- ***const*** keyword.

Connection matrix (e2n- Sparse)

- n2e = $e2n^T$ - node to element matrix

- n2n=n2e*e2n – node to node matrix



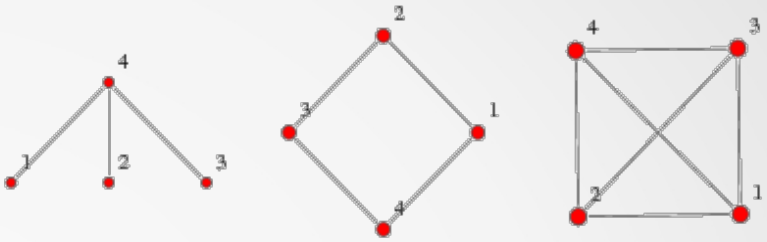
$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

- Matrix with rows and columns labeled by graph vertices, with a 1 or 0 in position according to whether and are adjacent or not.
- For a simple graph with no self-loops, the adjacency matrix must have 0s on the diagonal.
- For an undirected graph, the adjacency matrix is symmetric.

Connection matrix (e2n- Sparse)

- n2e = $e2n^T$ - node to element matrix

- n2n=n2e*e2n – node to node matrix



$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

- Matrix with rows and columns labeled by graph vertices, with a 1 or 0 in position according to whether and are adjacent or not.
- For a simple graph with no self-loops, the adjacency matrix must have 0s on the diagonal.
- For an undirected graph, the adjacency matrix is symmetric.

HyperCube1000.0 example

5
Tt 0 1 2 4 1
Tt 2 1 3 7 1
Tt 1 4 5 7 1
Tt 1 2 4 7 1
Tt 4 2 6 7 1

e2n =

1	1	1	0	1	0	0	0
0	1	1	1	0	0	0	1
0	1	0	0	1	1	0	1
0	1	1	0	1	0	0	1
0	0	1	0	1	0	1	1

n2e = e2n'

1	0	0	0	0
1	1	1	1	0
1	1	0	1	1
0	1	0	0	0
1	0	1	1	1
0	0	1	0	0
0	0	0	0	1
0	1	1	1	1

```
n2eCol=[0 0 1 2 3 0 1 3 4 1 0 2 3 4 2 4 1 2 3 4]
n2eRow= [1 4 4 1 4 1 1 4]
N2eDsp = [0 1 5 9 10 14 15 16 20]
```

```
void updateScheme::getNeighborsOf(const int& x ,vector<int>& nbhs){

    for(int i=getN2nRow()[x];i<getN2nRow()[x+1];i++){

            if(getN2nCol()[i] != x) nbhs.push_back(getN2nCol()[i]);
    }

}
```

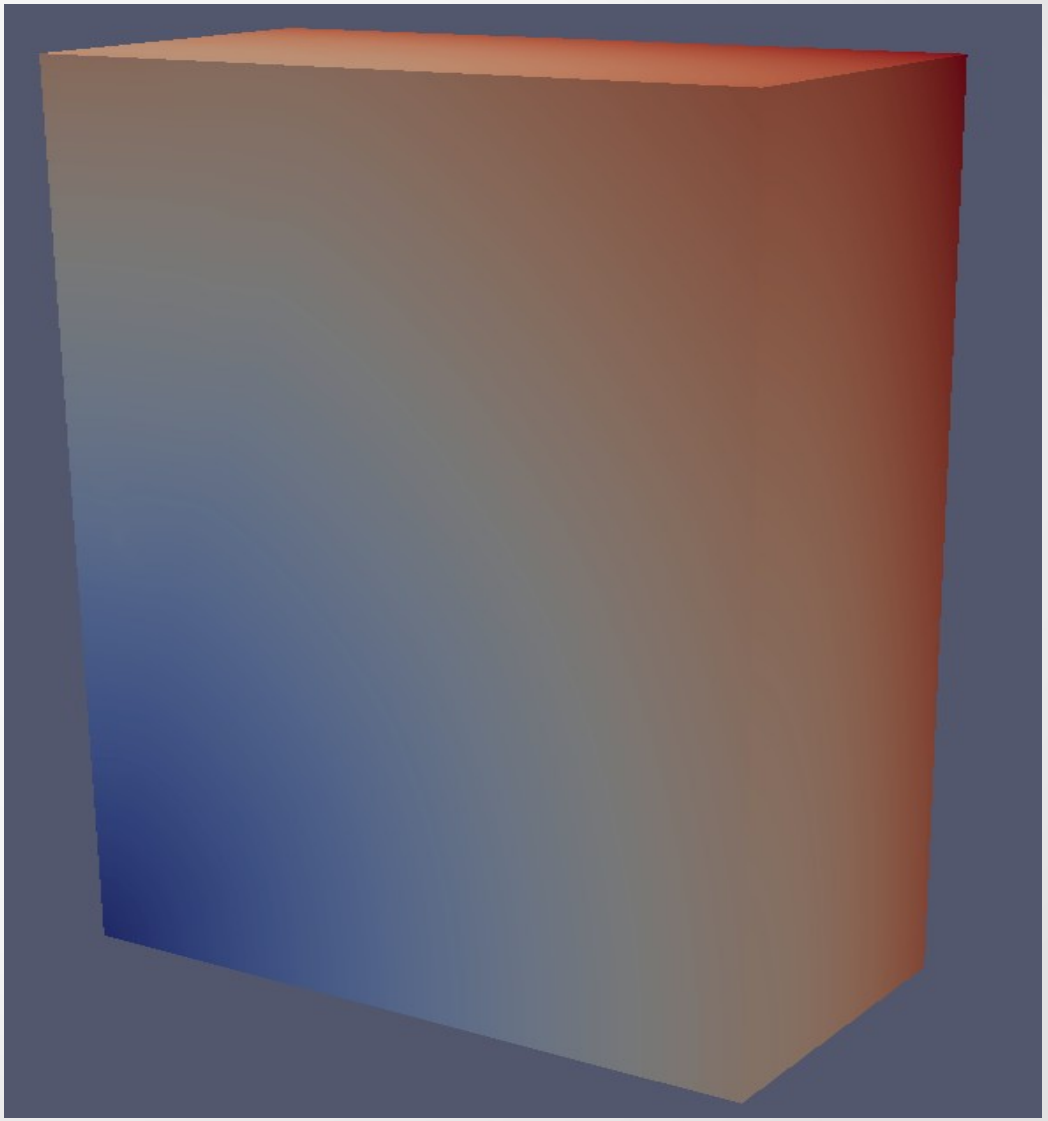
Results

- slab file:

  240000 tetrahedrons

  52521 vertices

- Iterative update time (**no optimization**) 250 sec
- Iterative update time optimized – 15 sec



•Pragma omp critical **not** optimized

```
void updateScheme::process(const int & list_iter,set<int>& activeList){
      float p,d,q;
     #pragma omp critical
      {
         p= getU()[list_iter];
         q= minSolutionOneRingTetra(list_iter,p); //calculate the solution
         if(p>q){
                              getU()[list_iter] = q;
             }
     }
         if(abs(p-q)< epsilon*(1+(abs(p)+abs(q))/2)){ //if converges then
```

```
        if(idx != list_iter){
            if(activeList.find(getN2nCol()[i]) == activeList.cend()){
                    #pragma omp critical
                    {
                        p= getU()[idx];
                        d=getU()[list_iter];
                    if(p > d) {// add only down wind neighbors
                        q= minSolutionOneRingTetra(idx,p); // calculate t
                        if(p>q){ // if the new value computed is less tha
                                if(q<getU()[idx])
                                    getU()[idx]=q;
                            activeList.insert(idx); // insert the new ele
                        }
                    }
                }
            }
        }
    }
    activeList.erase(list_iter); //remove x from L
}
```

const keyword

```
vector<float>& getU(){
    return U;
}

const vector<float>& getU() const{
    return U;
}
```

- No need to use pragma *atomic* or *critical* for read access anymore!
- Increased performance!

```
float updateScheme::minSolutionOneRingTetra(int x,float minSolution) const{

    array<int,3> id;                                // avoids alloc/dealloc of memory

    for(int i=getN2eRow()[x];i<getN2eRow()[x+1];i++){ // for each nbh element of x compute the eikonal solution and get the minimal one

        auto& elem = getElem()[getN2eCol()[i]];

        copy_if(elem.cbegin(), elem.cend(), id.begin(), [&](int i){return i!=x;} );

        float phi0,phi1,phi2;

/*      #pragma omp critical
         {*/

//#pragma omp atomic read
        phi0=getU()[id[0]];

//#pragma omp atomic read
        phi1=getU()[id[1]];

//#pragma omp atomic read
        phi2=getU()[id[2]];
    //}
        float travel_time= solver.solve(getPts(),id,x,phi0,phi1,phi2,getM(),epsilon);

        if (travel_time < minSolution)
            minSolution = travel_time;
    }

    return minSolution;
}
```

```
const vector<float>& getU() const{
    return U;
}
Press 'F2' for focus
```

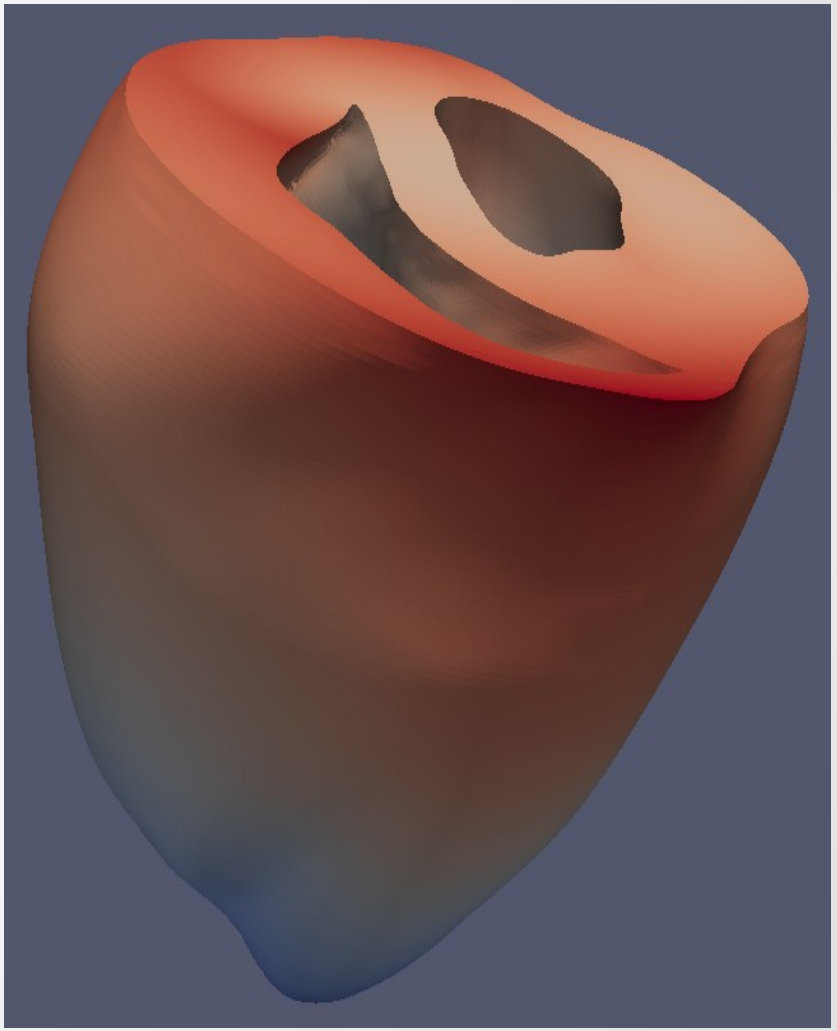
Results

- TbunnyC_i file:

  3073529 tetrahedrons

  547680 vertices

- Iterative update time 60.8668 sec
- Parallel update time (**no optimization**) 103.433 sec
- Parallel update time optimized 16.8859 sec



Intel Hyper-Threading Technology

- In practice we take full advantage of hyper-Threading tech.
- We choose N to be twice the number of CPU cores.
- Number of threads is equals to the number of sublists.
- To each thread a sublist is assigned for processing.

Results for TbunnyC_i:

- Number of threads equal to the number of sublists and processor cores (4). Update time – 25.9162 sec

- Number of threads equal to the number of sublists twice the number of processor cores (4). Update time – 18.1209 sec

•G++,clang++ and Intel compiler

- G++
- g++ -I"/home/daniel/git/eeikonalc/src/headers" -std=c++11 -O3 -Wall -c -g -pedantic -Wextra -Wshadow -Wconversion -Weffc++ -fopenmp EEikonalC++.cpp

- Clang++-3.6
- clang++-3.6 -c -g -Iheaders -Ofast -std=c++11 -Wheader-hygiene -Wconversion -Wshadow -Wextra -Wold-style-cast -Wshorten-64-to-32 -Wconditional-uninitialized -ferror-limit=3 -std=c++11 -fopenmp EEikonalC++.cpp

Results

- G++

1. [-Wshadow] warings

/ home/daniel/git/eeikonalc/src/headers/updateScheme.h : In member function ‘void updateScheme::setPts(std::vector<std::vector<float> >&)’:

/ home/daniel/git/eeikonalc/src/headers/updateScheme.h :66:43: warning: declaration of ‘pts’ shadows a member of 'this' [-Wshadow]

```
  void setPts(vector<vector<float> >& pts) {
                                      ^
```

2. [-Wconversion] Warnings

/ home/daniel/git/eeikonalc/src/headers/updateScheme.h :51:24: warning: conversion to ‘float’ alters ‘double’ constant value [-Wconversion]

```
  const float epsilon = 1e-5;
                    ^
```

- Clang++-3.6

1. [-Wshadow] warings

headers/updateScheme.h:66:38: warning: declaration shadows a field of 'updateScheme' [-Wshadow]

```
void setPts(vector<vector<float> >& pts) {
                                   ^
```

headers/updateScheme.h:33:25: note: previous declaration is here

```
vector<vector<float> > pts;
                     ^
```

2. [-Wconversion] Warnings

headers/updateScheme.h:51:24: warning: implicit conversion loses floating-point precision: 'double' to 'const float' [-Wconversion]

```
const float epsilon = 1e-5;
                  ~ ^~~~
```

Reasons why to use Clang++

1. More Helpful Diagnostics Messages

2. Reporting more warnings for the same flag

3. Thread Safety Analysis (clang++-3.7)
    1. Supports pthreads
    2. Mutex lock and unlock

Data race analysis

- Intel Inspector

updateScheme.cpp | Disassembly (a.out!0xba1f)

```
200                             if(p>q){ // if the new value computed is less than
201
202                                 #pragma omp critical
203                                 {
204
205                                     if(q<getU()[idx])
206
207                                         //#pragma omp atomic write
208                                         getU()[idx]=q;
209
210                                 }
211
212                                 activeList.insert(idx); // insert the new eleme
213                             }
214                         }
215
216
217                     }
```

Read - Thread OMP Worker Thread #1 (10705) (a.out!process - updateScheme.cpp:191)

updateScheme.cpp | Disassembly (a.out!0xb97f)

```
183                 const int idx=getN2nCol()[i];
184
185                 if(idx != list_iter){
186
187                     if(activeList.find(getN2nCol()[i]) == activeList.cend()){
188  /*
189                         #pragma omp critical
190                         {*/
191                             p= getU()[idx];
192
193                             d=getU()[list_iter];
194                         //}
```

- Clang++-3.6

  Compiled with -Wthread-safety flag!

  - No result!

•Android NDK

•NDK is a toolset that allows you to implement parts of your app using *native-code (jni)* languages such as C and C++

•Use the NDK if your app is truly processor bound—never because you simply prefer to program in C/C++

Android platform compatibility	
Native Code CPU Architecture Used	Compatible Android Platform(s)
ARM, ARM-NEON	Android 1.5 (API Level 3) and higher
x86	Android 2.3 (API Level 9) and higher
MIPS	Android 2.3 (API Level 9) and higher

To ensure compatibility,
an application using NDK must declare a ***<uses-sdk>*** element in its manifest.xml file ***android:minSdkVersion*** attribute value of "3" or higher. For example:

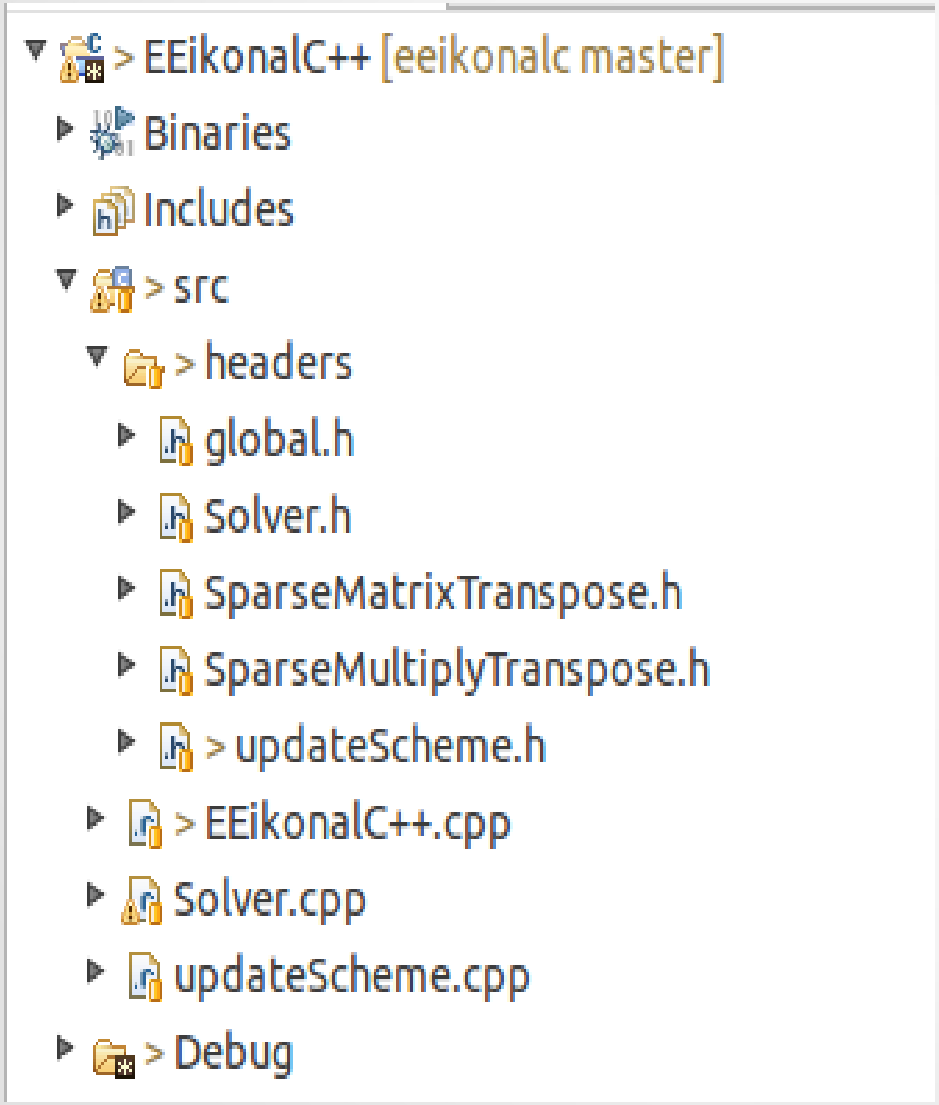
```
<manifest>
  <uses-sdk android:minSdkVersion="3" />
  ...
</manifest>
```

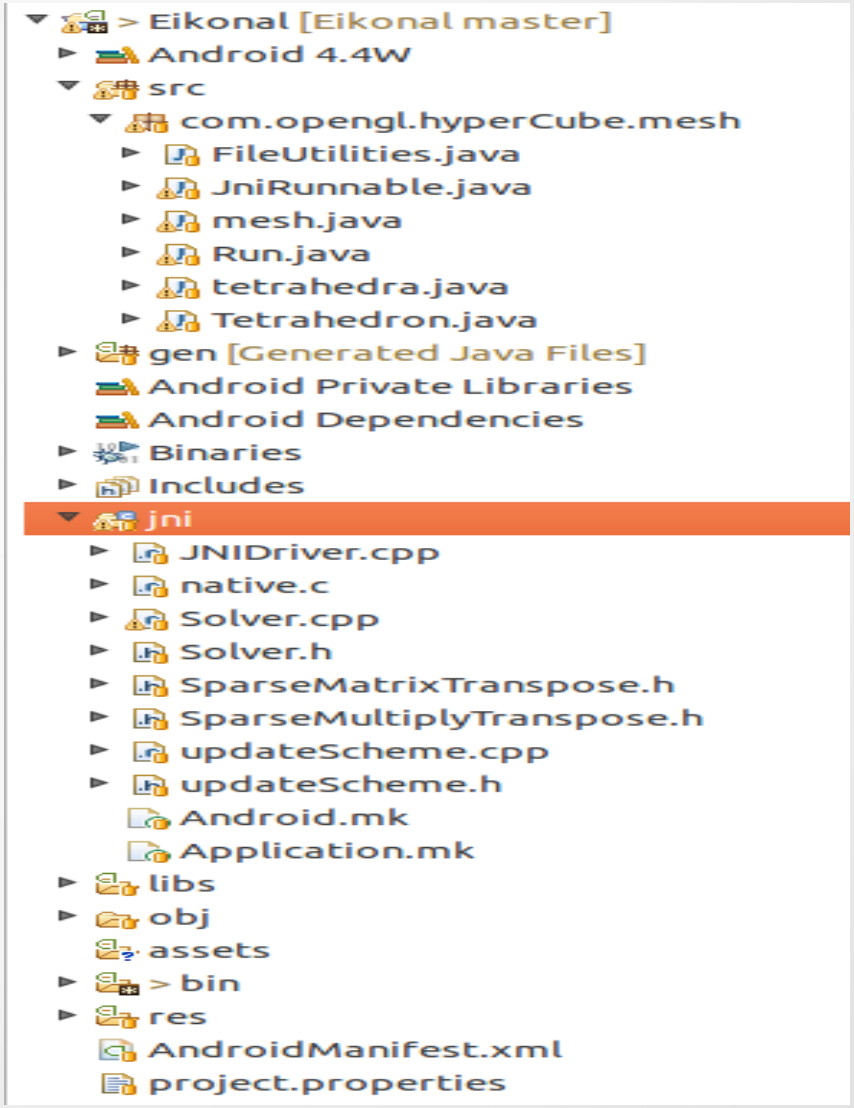
Getting Started with the NDK

1. Download and Install NDK package.
   - ndk$ chmod a+x android-ndk-r10c-darwin-x86_64.bin
   - ndk$ ./android-ndk-r10c-darwin-x86_64.bin
   - When uncompressed the NDK files are contained in a direcetory called android-ndk-<version> ready to use.

- Place your native sources under <project>/jni/...
- Create <project>/jni/Android.mk to describe your native sources to the NDK build system.
- Build your native code by running the 'ndk-build' script from your project's directory. It is located in the top-level NDK directory.
  - Cd <project>
  - Execute ndk-build* script
  - The build tools copy the stripped, shared libraries needed by your application to the proper location in the application's project directory.
- Finally, compile your application using the SDK tools in the usual way. The SDK build tools will package the shared libraries in the application's deployable .apk file.

Local C++ vs NDK C++ project





Jni Driver function

- The 'extern “c”' command tells the compiler to give the method “C linkage”
- Native C++ function
- Specific function name pattern “Java_package_class_method”
- Takes a *Java object string* and the *JNI environment* parameters
- Converts the Java String to C++ String
- Then it calls an ordinary C++ function

```
extern "C" { /*C++ function names get mangled by the
                compiler (to support overloading and other things) -- unless
                you specify extern "C".  If you forget the extern
                declaration, JNI will be unable to find the function
                implementations to match your Java "native" declarations*
                Important: Only for C++ code no need for C*/

    //“JNIEXPORT void JNICALL” defines the return type of the method as a Java-compatible string.
    JNIEXPORT void JNICALL
    Java_com_opengl_hyperCube_mesh_JniRunnable_solveEikonalOn(JNIEnv * env, jobject obj, jstring logThis){

        jboolean isCopy;

        const char * szLogThis = (env)->GetStringUTFChars(logThis, &isCopy);

        string elemFile = szLogThis;

        JNIDriver jd;

        jd.passControlfromJnitoEikonal(elemFile);

        (env)->ReleaseStringUTFChars(logThis, szLogThis);
    }
}
```

Android.mk and Application.mk

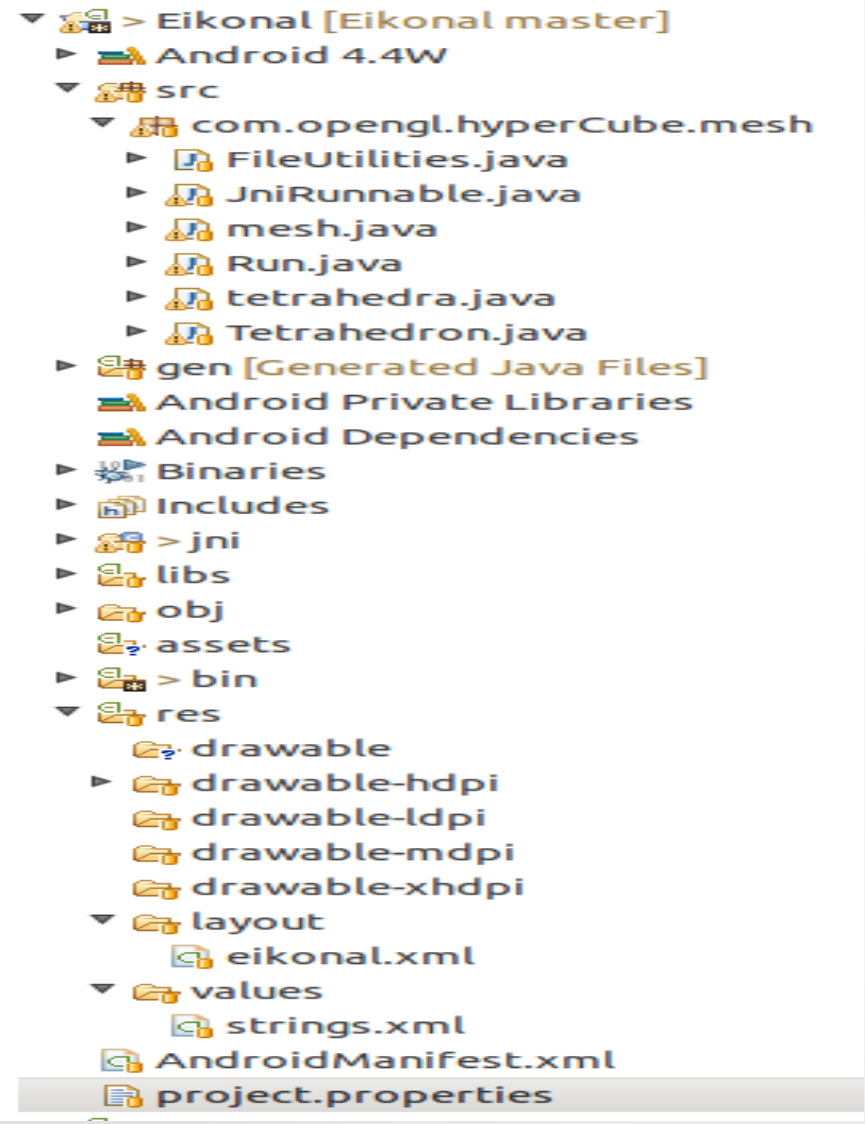
```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)

LOCAL_MODULE    := Eikonal
LOCAL_SRC_FILES :=  updateScheme.cpp JNIDriver.cpp Solver.cpp
LOCAL_LDLIBS := -llog
LOCAL_CFLAGS += -fopenmp
LOCAL_LDFLAGS += -fopenmp
LOCAL_EXPORT_C_INCLUDE_DIRS  := $(LOCAL_PATH)
include $(BUILD_SHARED_LIBRARY)
```

```
NDK_TOOLCHAIN_VERSION := 4.9
#APP_ABI := armeabi armeabi-v7a x86 mips
#  Enable C++11. However, pthread, rtti and exceptions aren't enabled
APP_CPPFLAGS += -std=c++11
# Instruct to use the static GNU STL implementation
APP_STL := gnustl_static
LOCAL_C_INCLUDES += ${ANDROID_NDK}/sources/cxx-stl/gnu-libstdc++/4.9/include
```

- Local module to be loaded in the Java part.
- Recommended to use android logging feature -llog (/system/lib/liblog.so).
- As of revision 9, the Android NDK comes with support for OpenMP.
- Important you need LDFLAGS also for the linker.

- The default compiler is GCC 4.6. The NDK_TOOLCHAIN_VERSION key specifies the 4.9 value to define the toolchain to use GCC 4.9 as the compiler.
- The default configuration of the NDK build system generates the machine code for the armeabi; that is, the ARMv5TE-based CPU. The APP_ABI value allows you to select different ABIs.
- The following line enables the supported C++11 features for all your modules
- GNU STL implementation for STL features
- The following line specifies the path to GNU STL implementation headers included in the NDK folders. Because of the specified toolchain use GCC 4.9, make sure the headers path corresponds to version 4.9

Android Project Eclipse



- Good reference how to create a Android project in Eclipse:
  - http://developer.android.com/tools/projects/projects-eclipse.html#SettingUpLibraryProject
- JniRunnable Thread class for handling the dialog job.
- Mesh: GlSurfaceView class for drawing tetrahedras using OpenGl.
- Run: the activity class.
- Tetrahedron class for creating tetrahedra objects to be drawn.
- FileUtilities class handling the file load from /sdcard.

Manifest.xml, String.sml, eikonal.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.opengl.hyperCube.mesh"
      android:versionCode="1"
      android:versionName="1.0">
<uses-sdk android:minSdkVersion="18"
    android:targetSdkVersion="22"/>

    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"  />
    <uses-permission android:name="android.permission.WRITE_SETTINGS" />

    <application android:icon="@drawable/icon" android:label="@string/app_name"
                 android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
                 android:allowBackup="false"
                 android:allowClearUserData="true"
                 android:hardwareAccelerated="true"
                 android:largeHeap="true">
        <activity android:hardwareAccelerated="true" android:name="com.opengl.hyperCube.mesh.Run"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>

</manifest>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Eikonal Solver</string>
    <string name="author">Daniel Ganellari</string>
    <string name="dialog_title">Open Data</string>
        <string-array
        name="mesh">
        <item>hyperCube100.0</item>
        <item>slab</item>
        <item>TBunnyC2_i</item>
        <item>TBunnyC_i</item>
    </string-array>

    <color name="blueAndroid">#0065B3</color>

</resources>
```

```xml
<?xml version="1.0" encoding="utf-8"?>

    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:background="@color/blueAndroid"
        android:id= "@+id/linearlayout1" >

    <com.opengl.hyperCube.mesh.mesh
        android:id="@+id/glSurfaceViewID"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="0.23" />


</LinearLayout>
```

Create the Activity Class

- The class should extend Activity class
- Declare the openGL view
- Declare the JniRunnable and start the new Thread.
  - Not mandatory! Just for needs of this app.
- SetContentView sets the layout declared in the eikonal.xml file and sets it as the view.
- Start the thread!
- An activity class should contain also onPause() and onResume() but they are automatically created after extension as override methods.

```
 *
 * @author Daniel Ganellari
 */
public class Run extends Activity {

    /** The OpenGL View */
    private mesh mes;

    public static JniRunnable meshRun =new  JniRunnable(false);

    /**
     * Initiate the OpenGL View and set our own
     * Renderer (@see Lesson05.java)
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

         setContentView(R.layout.eikonal);

         mes = (mesh) findViewById(R.id.glSurfaceViewID);

        if(meshRun.exists==false){// run the thread solver only

            meshRun.setContext(this);
            meshRun.start();
            meshRun.exists=true;
        }
    }
```

Android Parallel programming & Native call

```java
public class JniRunnable extends Thread{

    ProgressDialog progressDoalog;
    String fileName;
    Context context;
    boolean solve;
    boolean exists=false;

    public JniRunnable(boolean solve) {
        super();
        this.solve=solve;
    }

    private native void solveEikonalOn(String logThis);

    private void  solve(String filename){
        File sdcard = Environment.getExternalStorageDirectory();
        File file = new File(sdcard,filename);
        solveEikonalOn(file.getPath());
    }

    @Override
    public void run() {

        System.loadLibrary("Eikonal");
        // load here since jni load should not be done in different threads,
        //but only at the one who loads and declares the native function as well

        try {

            while (true) {
                // always running and performing solve eikonal whenever one file is chosen.

                if(solve){

                    solve(fileName);
                    // call the native jni function for eikonal solve

                    solve=false;
                    progressDoalog.dismiss();
                    ((Activity) context).setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_SENSOR);
                    // after the processing has finished unlock the rotation

                    mesh.render = true; // and now also start rendering the mesh into the screen.
                }
            }

        } catch (Exception e) {
            e.printStackTrace();
```

- For a class to be a runnable instance it should inherit from Thread class ore implement Runnable interface.
- Native function declaration same name as in the NDK C++ impl.
- Call the native NDK function *solveEikonalOn* inside the solve function.
- Load the “Eikonal” Library as declared into the Android.mk NDK file.
- The Thread runs during all the lifecycle and executes solve function whenever a file is chosen and solve variable is set to true.
- After the solving has finished the rendering of the mesh is done and the progress Dialog is dismissed.
- **Important**: *The native function declaration, library load and call should be done by the same Thread.*

OpenGL Surface View class

```java
@SuppressLint("ClickableViewAccessibility")
public class mesh extends GLSurfaceView implements Renderer {
```

```java
@Override
public boolean onTouchEvent(MotionEvent event) {
```

- Create an item list menu by using an alert dialog which reads the R.array.mesh created in the string.xml file.
- OnClick function the dialog is closed, a file name and index position chosen.
- The path is then built.
- A progress dialog is built and shown which is then closed when the Eikonal equation is solved for the chosen file.
- Properties for the JniRunnable static object are set which are read from the thread and upon solve=true it starts solving the file chosen.
- Then a requestRender() is invoked to start drawing the tetrahedra.

```java
// 1. Instantiate an AlertDialog.Builder with its constructor
AlertDialog.Builder builder = new AlertDialog.Builder(this.getContext());

// 2. Chain together various setter methods to set the dialog characteristics
builder.setTitle(R.string.dialog_title)
       .setItems(R.array.mesh, new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int which) {
    // The 'which' argument contains the index position          // of the selected item

        dialog.cancel();
        ((Activity) getContext()).setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LOCKED);

        String[] meshNames = getResources().getStringArray(R.array.mesh);
        String eikonalFile = "Eikonal/" + meshNames[which];

        FileUtilities.file=eikonalFile;

        progressDoalog = new ProgressDialog(getContext());
        progressDoalog.setMax(40);
        progressDoalog.setMessage("Please Wait.");
        progressDoalog.setTitle("Processing " + meshNames[which] + " ...");
        progressDoalog.setIndeterminate(true);

        progressDoalog.show();

        //For every chosen file tell the thread working on the processing to process the eikonalfile,
        //show the progressdialog and solve the equation.
        Run.meshRun.setFileName(eikonalFile);
        Run.meshRun.setProgressDoalog(progressDoalog);
        Run.meshRun.setSolve(true);
        Run.meshRun.setContext(getContext());


        requestRender();
        // request the rendering of the mesh into the screen after the processing is finished.
}

});

// 3. Get the AlertDialog from create()
AlertDialog dialog = builder.create();
dialog.show();
```

Results on Android

- slab file:

  240000 tetrahedrons

  52521 vertices

- Parallel update time optimized
  8.599 sec

- TBunnyC_i file:

  3073529 tetrahedrons

  547680 vertices

- Parallel update time optimized
  257.141 sec

- TBunnyC2_i file:

  266846 tetrahedrons

  59292 vertices

- Parallel update time optimized
  29.657 sec

- *Running the tests with*

  4 sublists 4 threads.

Conclusions

- Local C++ code giving very good results.
- Solving on parallel gives better results on solution values than iteratively since more solutions are found on Tetrahedrons than on Triangle surface sides or edges of the Tetrahedrons.
- Solving on Android gives satisfactory results.
- There is still room for performance improvement.
- Thinking on memory layout and vectorization.

References

- [1] W.-K. Jeong and R. T. Whitaker, A fast iterative method for eikonal equations, SIAM J. Sci. Comput., 30 (2008), pp. 2512–2534.

- [2] Z. Fu, W.-K. Jeong, Y. Pan, R. M. Kirby, and R. T. Whitaker, A fast iterative method for solving the eikonal equation on triangulated surfaces, SIAM J. Sci. Comput., 33 (2011), pp. 2468–2488.

- [3] A FAST ITERATIVE METHOD FOR SOLVING THE EIKONALEQUATION ON TETRAHEDRAL DOMAINS ∗ZHISONG FU , ROBERT M. KIRBY , AND ROSS T. WHITAKER SIAM J. S CI. C OMPUT .Vol. 35, No. 5, pp. C473–C494