ACTA UNIVERSITATIS AGRICULTURAE ET SILVICULTURAE MENDELIANAE BRUNENSIS
SBORNÍK MENDELOVY ZEMĚDĚLSKÉ A LESNICKÉ UNIVERZITY V BRNĚ

Volume LVII 8 Number 1, 2009

3D VISUALIZATION AND FINITE ELEMENT MESH FORMATION FROM WOOD ANATOMY SAMPLES PART II – ALGORITHM APPROACH

P. Koňas

Received: October 14, 2008

Abstract

KOŇAS, P.: *3D visualization and finite element mesh formation from wood anatomy samples, Part II – Algorithm approach.* Acta univ. agric. et silvic. Mendel. Brun., 2009, LVII, No. 1, pp. 79–88

Paper presents new original application WOOD3D in form of program code assembling. The work extends the previous article "Part I – Theoretical approach" in detail description of implemented C++ classes of utilized projects Visualization Toolkit (VTK), Insight Toolkit (ITK) and MIMX. Code is written in CMake style and it is available as multiplatform application. Currently GNU Linux (32/64b) and MS Windows (32/64b) platforms were released. Article discusses various filter classes for image filtering. Mainly Otsu and Binary threshold filters are classified for anatomy wood samples thresholding. Registration of images series is emphasized for difference of colour spaces compensation is included. Resulted work flow of image analysis is new methodological approach for images processing through the composition, visualization, filtering, registration and finite element mesh formation. Application generates script in ANSYS parametric design language (APDL) which is fully compatible with ANSYS finite element solver and designer environment. The script includes the whole definition of unstructured finite element mesh formed by individual elements and nodes. Due to simple notation, the same script can be used for generation of geometrical entities in element positions. Such formed volumetric entities are prepared for further geometry approximation (e.g. by boolean or more advanced methods). Hexahedral and tetrahedral types of mesh elements are formed on user request with specified mesh options. Hexahedral meshes are formed both with uniform element size and with anisotropic character. Modified octree method for hexahedral mesh with anisotropic character was declared in application. Multicore CPUs in the application are supported for fast image analysis realization. Visualization of image series and consequent 3D image are realized in VTK format sufficiently known and public format, visualized in GPL application Paraview. Future work based on mesh improvement through mesh error statistic, image acquisition and thresholding improvement by more sophisticated filters together with code optimization for fast image analysis is discussed. Also fractal characteristics classification on microscopic scale level is taken into account for further work.

wood, anatomy, binary image, finite element, mesh converter

Mapping of pixels intensity values from space of fixed image into the moving image is made by class of linear interpolator itk::LinearInterpolateImage Function, which evaluates intensity values on non-grid values of moving image, which is generally deformed according to the fixed image.

In presented program two thresholding filters were used. By utilization of ITK two following filters were included. Otsu filter based on itkOtsuThresholdImageFilter class and binary filter based on itkBinaryThresholdImageFilter were used. Both of thresholding filters process image according to appropriate threshold. Whereas Otsu filter automatically computes value of threshold, binary filter allows defining the user value for sensitive separation of structure from image background.

Mean value of pixels within the selected region is computed by itk::MeanCalculator and itk::CovarianceCalculator classes.

Contouring is made by another class (itkSimpleContourExtractorImageFilter). Objects of this class mask pixels of image on interface of structure and image background. Precisely, it selects pixels which are within the set of the structure (the last boundary). Although the class offers definition of radius of interest (and this way the size of kernel for convolution), only one pixel of neighbourhood is taken into account, because of shortening of computing time.

Meshing is realized by vtkDelaunay3D class, which triangulates the significant points of the image and forms unstructured grid of tetrahedral elements. Similar approach, but not purely in ITK has been realized in project of tetrahedral mesh generation for medical imaging (Fedorov, 2005). This work is probable one of the most important works in OpenSource field for quality tetrahedral mesh generation from image series. Unfortunately project consists of large amount of source code from various authors which leads to incompatibility with modern form of ITK and VTK. Code is also focused on MR images and not for series of separate 2D images. Big portion of code use the project PETSc which is not CMake based and lot of difficulties with platform dependent code makes almost impossible to generalize the code into the pure ITK platform independent program.

Mesh with good quality provides implemented code of huge project MIMx. This code forms regular unstructured hexahedral rectilinear mesh with prescribed size of elements. Also modulus of elasticity can be computed on base of image intensity and input estimation of Young modulus of the structure. This part of code was developed in MIMx project and described in Carter 1977.

MATERIALS AND METHOD

Code was written in KDevelop and Microsoft Visual C++ IDE GUI environment. Both Linux and MS Windows platform were used for compiling and testing of code. Multiplatform code is based on project CMake (Martin, 2005). Classes of Insight toolkit (ITK) (Yoo et al., 2002) and Visual toolkit (VTK) projects were used for assembling the code. Very useful source for our work was Ibáñez 2005 which provided a lot of basic and advantage code for image processing and Prata 2004 which was useful for implementation into C++ code. Final binary file is small and standalone without any linked libraries. Processing of images for tetrahedral meshes with prefiltration and registration takes app. 1.5 hour. Whereas the most time consuming part is registration of full images (1 hour) and tetrahedralization (30 minutes).

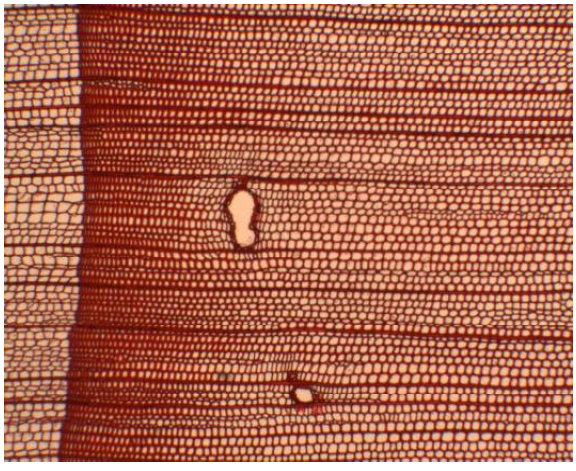
RESULTS AND DISCUSSION

The application is in this phase without GUI and user has to specify parameters in command line for specific control. Numerous parameters allow detail control of program in all phases and almost in all available variables of individual object methods.

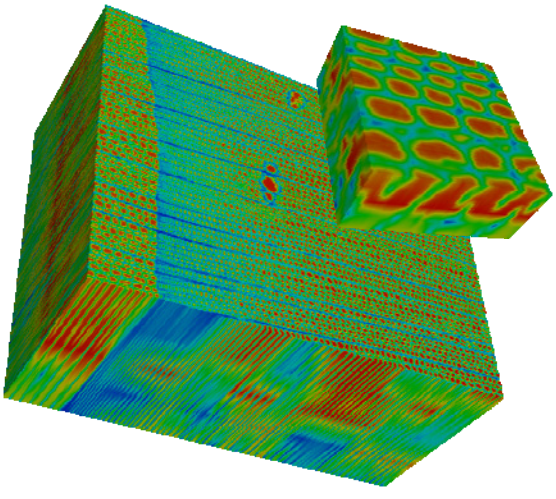
Program code is relatively huge (more than 1500 code lines). Partial description in the text is done by simplification and idealization of program code. Most of mentioned methods keeps C++ notation and style and should be also readable for users non skilled in programming. The following diagram represents the schematic flow of application for image analysis and final output of ANSYS APDL script (Fig. 3).

I: *Command line parameters of application WOOD3D*

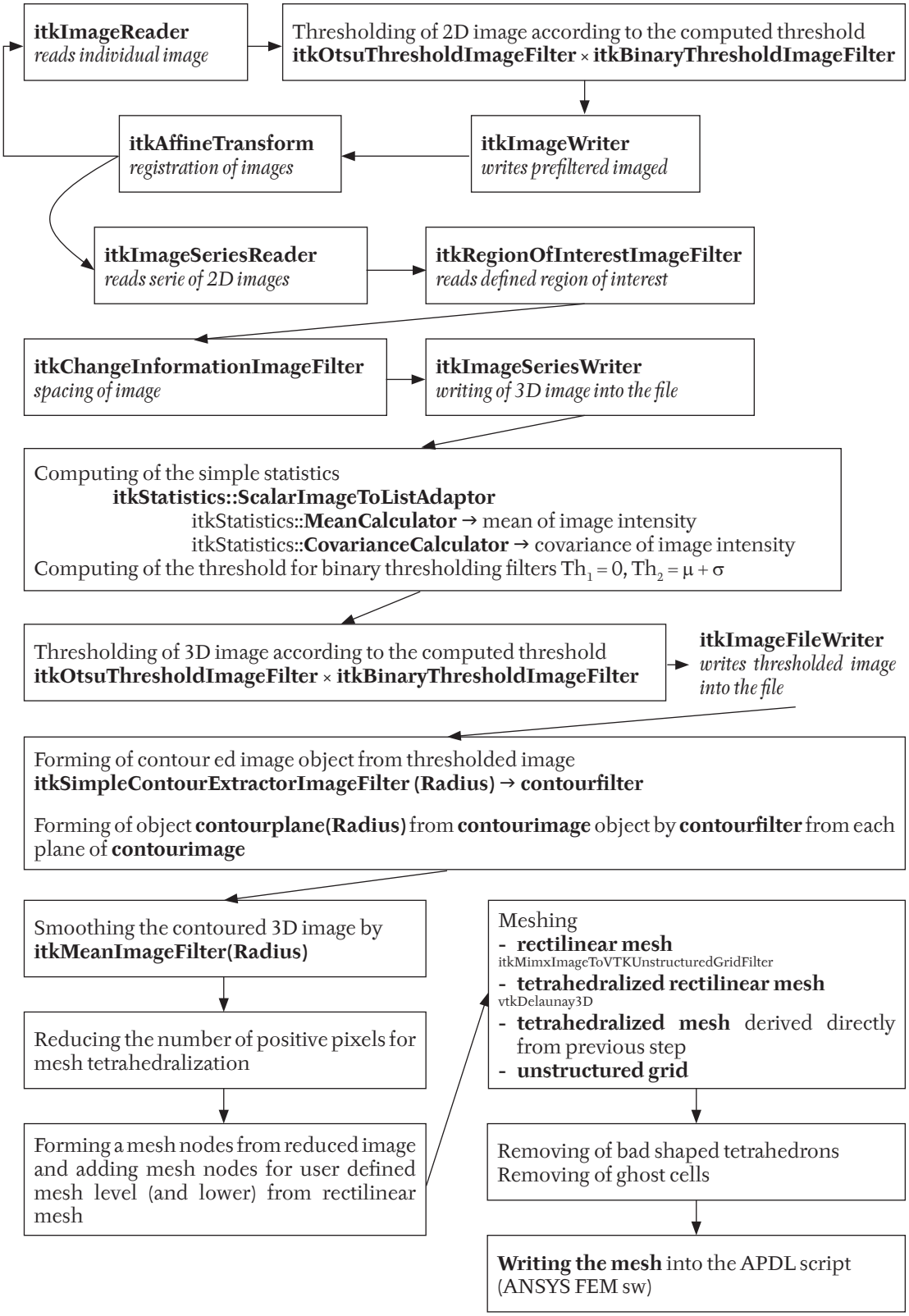
-first *N*	Number of the first image of series (it is assumed in format img00x.png). Usually it is 1
-last *N*	Number of the last image of series.
-file *filename*	Filename of output VTK file
-size *Sx Sy Sz*	Size of region subtracted from image. Size is defined as vector by three numbers in pixels. Default is full size of image.
-origin *Ox Oy Oz*	Origin for subtracted region. Origin is defined as vector by three numbers in pixels. Default is 0 0 0
-filter_lower *LowerIntensity*	Lower intensity value for binary thresholding filter. Default is 0.
-filter_upper *UpperIntensity*	Upper intensity value for binary thresholding filter. If no value is defined, then optimal value is automatically computed.
-filter_file *filename*	Filename for output VTK file of thresholded image.
-filter *N*	0=Otsu thresholding filter, 1=Binary thresholding filter
-mesh_size *Size*	Size of output elements (in pixels)
-scale *Cx Cy Cz*	Scaling coefficients. Scaling is defined as vector by three numbers. Default is 1 1 1
-tetra *0|1*	0=no tetrahedrons will be created 1=tetrahedrons are created
-tetra_type *0|1*	0=no tetras will be created from hexahedral mesh 1=tetras will be created from hexahedral mesh
-reduced *0|1*	0=reduction of pixels for tetrahedralization is performed 1=reduction is performed
-prefiltered *0|1*	0=no prefiltering of individual input images is performed 1=prefiltering will be realized
-smoothed *0|1*	0=smoothing with mean filter is performed 1=smoothing is performed
-registered *0|1*	0=individual images are not registestered 1=images are registered at the beginning
-alpha *N*	Size of element which is filtered out of mesh (in pixels)



1: *Input image of spruce wood*



2: *3D VTK image*



3: *Diagram of application flow*

Next itk::ImageSeriesReader is initiated for reading of the whole native/prefiltered and/or registered image series. Images are RGB type without any previous image preparation (Fig. 1). Image series is exported and saved into 3D VTK image format (Fig. 2). Registration is based on itk::ImageRegistrationMethod. It allows compute translation, rotation and scaling parameters for optimal images alignment (Code 1).

```
registration->SetMetric(        metric        );
registration->SetOptimizer(     optimizer     );
registration->SetInterpolator(  interpolator  );
TransformType::Pointer  transform = TransformType::New();
registration->SetTransform( transform );
fixedImageReader->SetFileName(  fixedfilename );
movingImageReader->SetFileName( movingfilename );
registration->SetFixedImage(    fixedImageReader->GetOutput()    );
registration->SetMovingImage(   movingImageReader->GetOutput()   );
fixedImageReader->Update();
typedef itk::CenteredTransformInitializer< TransformType, FixedImageType,
MovingImageType >  TransformInitializerType;
TransformInitializerType::Pointer initializer = TransformInitializerType::New();
initializer->SetTransform(   transform );
initializer->SetFixedImage(  fixedImageReader->GetOutput() );
initializer->SetMovingImage( movingImageReader->GetOutput() );
initializer->MomentsOn();
initializer->InitializeTransform();
registration->SetInitialTransformParameters( transform->GetParameters() );
typedef OptimizerType::ScalesType       OptimizerScalesType;
OptimizerScalesType optimizerScales( transform->GetNumberOfParameters() );
optimizerScales =  translationScale;
optimizer->SetScales( optimizerScales );
optimizer->SetMaximumStepLength( steplength );
optimizer->SetNumberOfIterations( maxNumberOfIterations );
optimizer->MinimizeOn();
registration->StartRegistration();
```

Code 1: *Registration of image series. Moving image is consequent image which is registered according to previous (fixed) image.*

Prefiltering is based on itk::OtsuThresholdImageFilter and itk::BinaryThresholdImageFilter. Otsu filter automatically computes value of threshold, binary filter allows defining the user values for sensitive separation of structure from image background (Code 2). Prefiltering is realized on individual images for unification of colour spaces.

```
if (filter_type == 0)
              {
              otsufilter->SetInput( onereader->GetOutput() );
              otsufilter->SetOutsideValue( 255);
              otsufilter->SetInsideValue(  0);
              otsufilter->Update();
              threshold = otsufilter->GetThreshold();
              } else
              {
                     binaryfilter->SetInput( onereader->GetOutput() );
                     binaryfilter->SetOutsideValue(255);
                     binaryfilter->SetInsideValue(0);
                     binaryfilter->SetLowerThreshold(filter_lower);
                     binaryfilter->SetUpperThreshold(filter_upper);
              }
```

Code 2: *Prefiltering of individual images by otsu and binary filter.*

Contouring processes the whole image. Nevertheless, individual plane regions are taken into account in this process and pixels forming the lumens boundary are marked for each plane separately. Contours are detected in vicinity of each pixels in radius of one pixel (Code 3). Processed image is written into the file.

```
for (int i=0; i<count_of_planes; i++)
 {
  planesize[0]=size[0]; planesize[1]=size[1]; planesize[2]=1;
  planestart[0]=0; planestart[1]=0; planestart[2]=i;
  Region.SetSize( planesize );
  Region.SetIndex( planestart );
  plane->SetInput( contourimage);
  plane->SetRegionOfInterest( Region );
  contour->SetInput(plane->GetOutput());
  contour->SetRadius(1);
  contour->Update();
  itk::ImageRegionIterator<ImageType> it(contour->GetOutput(), Region);
  for ( ; !it.IsAtEnd(); ++it)
       {
       pindex = it.GetIndex();
       pvalue=contour->GetOutput()->GetPixel(pindex);
       contour->GetOutput()->TransformIndexToPhysicalPoint(pindex,cpoint);
       cpoint[2]=i;
       isvalid=contourimage->TransformPhysicalPointToIndex(cpoint,ppindex);
       ppindex[2]=i;
       if (!isvalid) std::cout<<(bool) isvalid<<std::endl;
       if ((long) pvalue != 255) contourimage->SetPixel(ppindex,0);
         else contourimage->SetPixel(ppindex,255);
}
 }
```

Code 3: *Contouring of 3D image by planes*

The several virtual grids for each 2^n divider (as scale) of the image size are inserted through points of 3D image (Code 4). Mesh is based on points of contour filter, grid points and points from itkMeanImageFilter. The filter smears the contours of image. This way we can obtain non-binary pixel values in originally binary image which are good candidates for mesh points that well approximate the region around the edge of structure and contours in image respectively (Code 5). Formed codes are simpler and faster then similar body-centric cubic lattice algorithm (Molino et al., 2003).

```
for ( i = 1; i < mesh_levels = (log(max_grid_size)/log(2)); i++ )
       if ( pixel_value == is_positive_in_tresholded_image &&
              pixel_position == max_grid_size )
        {
          mesh_image -> SetPixel(pixel,1);
          vtk_point = pixel_position;
          points -> InsertNextPoint(vtk_point);
          mesh_grid_size = max_grid_size/2^i;
        }
```

Code 4: *Adding grid points into points of mesh*

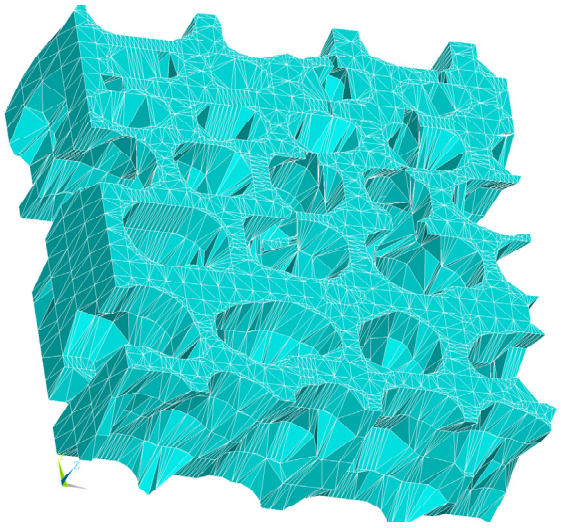
```
for ( i = 1; i < mesh_levels = (log(max_grid_size)/log(2)); i++ )
       if ( pixel_value == is_positive_in_tresholded_image &&
            pixel_value == is_nonzero_in_meanfilter &&
             pixel_position % mesh_grid_size == 0)
        {
          mesh_image -> SetPixel(pixel,1);
          vtk_point = pixel_position;
          points -> InsertNextPoint(vtk_point);
          mesh_grid_size = max_grid_size/2^i;
        }
```

Code 5: *Adding grid points near the edge of structure into points of mesh*

Points formed by the way mentioned above are triangulated by Delaunay algorithm with object of class vtkDelaunay3D (Fig. 4). The appropriate parameter alpha has to be defined for removing of elements with size bigger than alpha.

```
vtkDelaunay3D* tets=vtkDelaunay3D::New();
 tets->AddInput(points);
 if ( user_defined_alpha == 0)
          alpha=max_grid_size;
 tets->SetAlpha( alpha);
 tets->SetTolerance(1.0);
 tets->Update();
```

Code 6: *Tetrahedralization of selected points*



4: *Tetrahedralized mesh (48k elements) without tesssellation with prefiltering and registering*

Above objects generate a lot of bad shaped elements which are not appropriate for FE analysis. Therefore the mesh is tessellated by objects of vtkTessellatorFilter class. This class works directly on vtkUnstructuredGrid class and generates the same class on the output. It allowed simple implementation of filter for grid tessellation. Disadvantage of this old and temporary class (according to VTK Documentation project statement) is slow triangulation based on Delaunay method. As a possibility appears vtkGenericCellTessellator and vtkOrderedTriangulator (Shroeder, 2004) classes which offer very quick and parallelized/threaded algorithm for triangulation of reduced voxels together with definition of error metrics for finite elements which have to be tessellated.

As an alternative to Delaunay tetrahedralization the MIMx code for hexahedral mesh generation was also included. Advantage of MIMx code is also support of threads which accelerates run of application on multicored CPUs (Code 7).

```
imageToHexMeshFilter->SetInput( meanfilter->GetOutput() );
imageToHexMeshFilter->SetMaskImage(filtered_image);
imageToHexMeshFilter->SetMeshIndexOffset( 1 );
imageToHexMeshFilter->SetComputeMeshPropertiesOn( );
imageToHexMeshFilter->SetComputeMeshNodeNumberingOn( );
imageToHexMeshFilter->SetMeshResampleSize( mesh_size);
imageToHexMeshFilter->SetNumberOfThreads(n_core*n_cpu);
imageToHexMeshFilter->SetImageThreshold( threshold );
```

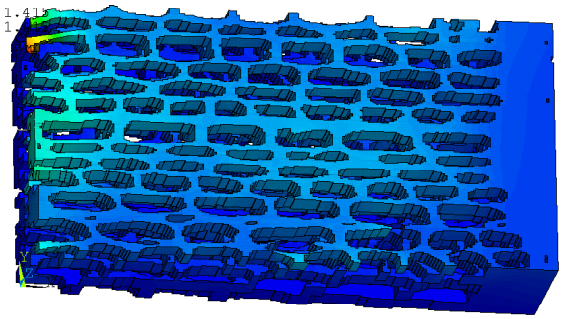
Code 7: *MIMx code for hexahedral mesh from image*

Third implemented code for meshing of image voxels forms unstructured grid with anisotropic irregular mesh similar to tetra mesh, but it is formed by divided cubes as in previous code. The mesh is created by declared and modified octree algorithm.

```
for (long i=0; (i<cubes_size); i++)
{
  cube_status=check_cube(cubes[i],otsufilter);
  if ( cube_status == 2 )
  {
   octet=divide_cube(cubes[i]);
   cubes[i].meshed=false;
   for (long k=0; k<octet.cubes_count; k++)
   {
          octet.cube[k].meshed=true;
          cubes[cubes_size+k]=octet.cube[k];
   }
   cubes_size=cubes_size+octet.cubes_count;
   Tcube * exp_cubes=new Tcube[cubes_size];
   for (long k=0; k<i; k++) exp_cubes[k]=cubes[k];
   long velikost_cubes=sizeof cubes / sizeof(Tcube);
   for (long k=i; k<velikost_cubes-1; k++) exp_cubes[k]=cubes[k+1];
   for (short j=0; j<=octet.cubes_count;j++)
      exp_cubes[velikost_cubes+j]=octet.cube[j];
   delete [] cubes;
   Tcube * cubes=new Tcube[cubes_size];
   for (long k=0; k<cubes_size; k++) cubes[k]=exp_cubes[k];
   delete [] exp_cubes;
  } else if (cube_status == cell)
   {
       cubes_count++; mesh[cubes_count]=cubes[i];
   }
}
```

Code 8: *Declared method of dividing cube*

Both Delaunay method and octree or dividing cube method form finite element mesh which is written into the ANSYS APDL script and allows forming of the mesh of finite elements in ANSYS application environment. Low scale offers big advantage in definition of wood structure as isotropic. This way, very accurate response in FE analysis can be obtained (in proposes that appropriate isotropic material properties are used). For test purposes structural problem was assembled with compression type of loading in longitudinal direction (in direction of cells length). Response in appearance of von misses equivalent for our case reveal low gradients and relative continuous stress distribution (Fig. 5).



5: *Stress field on FE model formed by MIMx hexahedral mesh and loaded by compression in transverse direction. Source image was not registered and prefiltered.*

CONCLUSION

Application WOOD3D is original output for image analysis which combines several program techniques of public code under GPL license from several projects together with several modifications typical for image analysis of wood anatomy samples. Usefulness of code was proved on real anatomy samples which were formed into 3D VTK visualization object by several image filters utilization (prefiltering, registering) and final region dividing by Delaunay tetrahedralization, octree method and dividing cube method applying. Final mesh offers user defined parameter control. Future work will be focused on implementation of advanced tessellation methods together with mesh error criterion definition for mesh quality improvement. Although a lot of different image filters was applied for wood structure emphasizing, no unique way was found. Nevertheless, code is sufficiently robust for soft woods as spruce e.g. with simple and regular structure. For hard woods more combinations of input filter parameters have to be used for appropriate image processing. Even that, no comparable pure or clear images as soft woods can be obtained. Enumeration of geometry of wood structure will be utilized in further works for microstructure fractal characteristics evaluation such as Hausdorff dimension by implementation and modification of itk::HausdorffDistance image filter.

SOUHRN

3D vizualizace a tvorba konečně prvkové sítě z anatomických vzorků dřeva,
Část II – Algoritmický přístup

Článek představuje novou původní aplikaci WOOD3D v jejím sestaveném programovém kódu. Práce rozšiřuje předchozí díl „Part I – Theoretical approach" detailním popisem implementovaných tříd C++ použitých projektů Visualization Toolkit (VTK), Insight Toolkit (ITK) and MIMX. Kód je napsán ve formátu CMake a je tak dostupný jako multiplatformní aplikace. Aktuálně byl uvolněn program pod platformami GNU Linux (32/64b) a MS Windows (32/64b). Článek diskutuje různé třídy filtrů zejména pro prahování obrazu. Je zdůrazněn význam procesu registrace pro kompenzaci rozdílnosti barevných prostorů řady vstupní obrázků. Výsledný vývojový diagram implementované obrazové analýzy je novým metodologickým přístupem pro zpracování obrazu v oblasti sestavení, vizualizace, filtrace, registrace a formování sítě konečných prvků. Aplikace vytváří skript parametrického jazyku ANSYS parametric design language (APDL), který je plně kompatibilní s prostředím konečně prvkového řešiče a modeláře programu ANSYS. Tento skript obsahuje úplnou definici nestrukturované konečně prvkové sítě tvořené jednotlivými elementy a uzly. Díky jednoduché notaci může být skript použit pro generování geometrických entit na místo elementů. Takto vytvořené objemové entity jsou připravené pro další aproximaci geometrie (např. booleovskými či jinými pokročilými metodami). Sítě šestistěnů jsou formovány jak v podobě rovnoměrné, tak s anizotropním charakterem. V aplikace je rovněž deklarována modifikovaná metoda octree pro tvorbu anizotropní sítě šestistěnů. Pro rychlé provedení obrazové analýzy podporuje aplikace vícejádrová CPU. Vizualizace řady vstupních obrázků a následný 3D model jsou realizovány ve formátu VTK (dostatečně známém a veřejném formátu) vizualizované v GPL aplikaci Paraview.
Je diskutováno i zaměření budoucí práce, která bude soustředěna na zlepšení kvality sítě pomocí statistik chyb sítě, pořízením obrázků a zlepšením pragovácích technik sofistikovanějšími filtry společně s optimalizací kódu pro rychlou obrazovou analýzu. Rovněž je vzato do úvahy budoucí využití aplikace pro klasifikaci fraktálových charakteristik na mikroskopické úrovni.

dřevo, anatomie, binární obraz, konečné prvky, převodník sítě

The Research project GP106/06/P363 Homogenization of material properties of wood for tasks from mechanics and thermodynamics (Czech Science Foundation) and Institutional research plan MSM6215648902 – Forest and Wood: the support of functionally integrated forest management and use of wood as a renewable raw material (2005–2010, Ministry of Education, Youth and Sport, Czech Republic) supported this work. This work benefited from the use of the Insight Segmentation and Registration Toolkit (ITK), open source software developed as an initiative of the U. S. National Library of Medicine.

REFERENCES

CARTER, D. R., HAYES, W. C., 1997: The compressive behavior of bone as a two-phase porous structure, *J of Bone Joint Surgery*, 59A: 954–962.

FEDOROV, A., CHRISOCHOIDES, N., KIKINIS, R., WARFIELD, S., 2005: Tetrahedral mesh generation for medical imaging, *http://hdl.handle.net/1926/35*

ITK, 2006: The Insight Segmentation and Registration Toolkit (ITK), www.itk.org

IBÁÑEZ, L., SCHROEDER, W., NG, L., CATES, J., 2005: *The ITK Software Guide – Second Edition Updated for ITK version 2.4*, Kitware, Inc., 804 p., ISBN 1-930934-15-7

MARTIN, K., HOFFMAN, B., 2005: *Mastering CMake – A Cross-Platform Build System*, Kitware, Inc., USA, 250 p.

MOLINO, N., BRIDSON, R., TERAN, J., FEDKIW, R., 2003: A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. *In: Proceedings of the 12th International Meshing Roundtable*, 103–114.

PRATA, S., 2004: *Mistrostvi v C++*, Computer Press, Brno, 1006 p.

SCHROEDER, W. J., GEVECI, B., MALATERRE, M., 2004: Compatible Triangulations of Spatial Decompositions. *In: Proceedings of Visualization 2004*, IEEE Press.

VTK, 2007: VTK Documentation Project (VTK 5.1.0 Documentation), www.vtk.org

YOO, T. S., ACKERMAN, M. J., LORENSES, W. E., SCHROEDER, W., CHALANA, V., AYLWARD, S., METAXES, D., WHITAKER, R., 2002: Engineering and Algorithm Design for an Image Processing API: A Technical Report on ITK – The Insight Toolkit. *In: Proceedings of Medicine Meets Virtual Reality, J. Westwood*, ed., IOS Press Amsterdam, pp. 586–592.

Address

Ing. Petr Koňas, Ph.D., Ústav nauky o dřevě, Mendelova zemědělská a lesnická univerzita v Brně, 613 00 Brno, Česká republika, e-mail:konas@mendelu.cz