Robust Exchange of Cartographic Models for Buildings and Roads: The CMU MAPSLab Site Exchange Format

George Bulwinkle Steven Douglas Cochran
J. Chris McGlone David McKeown Jefferey Shufelt

November 13, 1998
CMU–CS–98–134
(API Version 5.0)

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213–3890

Abstract

This report defines the CMU MAPSLab Site Exchange Format. This exchange format serves as a system independent method for import and export of three dimensional object descriptions to and from non-CMU systems. The Site Exchange Format supports road models, surface models, constraint models, and several types of building models.
This report also describes an application programmers interface to support this exchange format. This API has been implemented to allow users access to all components of the site model interchange structures, or any subset, for integration into any user's particular internal representation for object models.

This work was sponsored by the Advanced Research Projects Agency under Contract DACA76–95–C–0009 monitored by the U.S. Army Topographic Engineering Center, Fort Belvoir, VA. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Advanced Research Projects Agency, the U.S. Army Topographic Engineering Center, or the United States Government.

The Digital Mapping Laboratory's WWW Home Page may be found at: `http://www.cs.cmu.edu/ MAPSLab`

Keywords: building and road models, data exchange

Contents

Robust Exchange of Cartographic Models for Buildings and Roads: The CMU MAPSLab Site Exchange Format

George Bulwinkle, Steven Douglas Cochran, J. Chris McGlone, David McKeown, and Jefferey Shufelt

Digital Mapping Laboratory[1]
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213–3890

1 Introduction

This document defines the CMU MAPSLab Site Exchange Format. This exchange format serves as a system independent method for import and export of three dimensional object descriptions to and from non-CMU systems. The Site Exchange Format supports road models, road intersection models, surface models, constraint models, and several types of building models.

This document also describes an application programmers interface (CMU-SITE-API) to support this exchange format. This API has been implemented to allow users access to all components of the site model interchange structures, or any subset, for integration into any user's particular internal representation for object models. This API is documented in Section 8 and is available as a set of C language libraries. Four complete examples of interchange files are included in Section 9 for actual objects measured from the RADIUS modelboard, Fort Hood, and ISPRS Avenches test datasets.

We have developed several systems for automated building analysis within the MAPSLab. Three of these systems, VHBUILD [3], SiteCity [2], and PIVOT [6] have been modified to output three-dimensional object descriptions organized as site models as defined by this interchange format.

The significant design goals of this interchange format include:

- An easily readable interchange, in the form of an ASCII text file.
- The ability to specify the camera/sensor model for single or multiple images utilizing existing RADIUS standard interchange formats: USATEC header files and FBIP interchange files.
- The preservation of image identification as well as image measurements associated with model points for all images from which the model was constructed. The motivation is to provide for interchange of measured points allowing each user the ability to rigorously modify and regenerate the site model when new images or other information becomes available.
- Specification of the relationship of all vertices in the building object instance to the topology of the model to allow for unambiguous labeling.
- Rigorous association of the model to a local geodetic coordinate system, not to a graphic visualization system. This allows for an unambiguous single transformation that simplifies the implementation of the data exchange reader/interface and may lead to improved correctness, since there are fewer transformations to perform.
- The ability to optionally represent and exchange covariance and error information associated with object models to support the integration of these models with new or existing information about a site and to make estimates of model precision and quality.

[1]The Digital Mapping Laboratory's WWW Home Page may be found at: `http://www.cs.cmu.edu/~MAPSLab`

- Support for a small number of common shapes with the capability to support fully generic building objects.
- Avoid syntactic verbosity in the exchange definition, but provide some redundancy so that commonly used model parameters do not have to be calculated from primitive features.

For completeness we have incorporated the current description of the ASCII USATEC header interchange standard in Section 10 and the ASCII FBIP interchange format in Section 11. As other sensor models, such as RPC, become supported, using standard ASCII interchange formats these will be incorporated as well. We believe that the appropriate mechanism to provide sensor model specification is to reference standard models rather than developing new variations and incorporating them into the CMU MAPSLab Site Exchange Format.

Our research on the sitexchange format was supported by the Defense Advanced Research Projects Agency (DARPA/ISO) under Contract DACA76-95-C-0009 and is monitored by the U.S. Army Topographic Engineering Center. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency, the U.S. Army Topographic Engineering Center, or of the United States Government.

1.1 Usage and distribution disclaimer

The software described in this manual is the by-product and not the primary focus of our program of research. No warranty, either expressed or implied, is provided by Carnegie Mellon University for the use of this software. All rights are reserved, including commercial license, by the Digital Mapping Laboratory, Carnegie Mellon University, Pittsburgh, PA.

Distribution of this software is to the U.S. Government to support test and evaluation. Permission for its use must be secured by contacting Mr. Doug Caldwell, U.S. Army Topographic Engineering Center, (caldwell@tec.army.mil). The DARPA point of contact for this research is Mr. George Lukes, Program Manager for Image Understanding, DARPA/ISO (glukes@darpa.mil). The MAPSLab point of contact for questions regarding this software and its distribution is Dave McKeown (dmm@cs.cmu.edu).

Any redistribution of this document and associated software must contain this disclaimer, without alteration.

2 Overview

2.1 Images and Building Types

Figure 1 shows the image coordinate system used at the MAPSLab and defined by this interchange format. The image origin, (0,0) is taken to be at the upper left-hand corner of the raster, with rows increasing down along the vertical axis and columns increasing across the horizontal axis. For an image I of size N rows by M columns the pixel address in the lower right hand corner has the index of I(N-1, M-1).

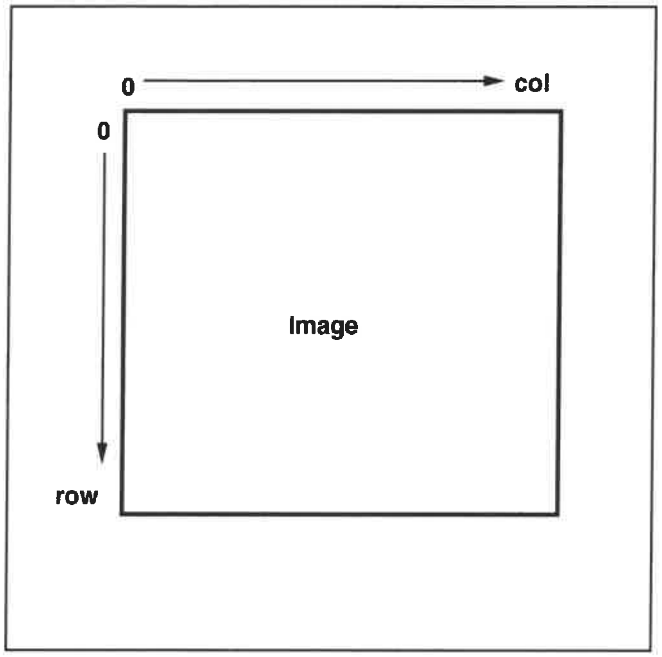

Figure 1: Image Coordinate System

Figure 2 shows the relationship between the five different types of building models currently supported by this interchange format. The building models can be classified hierarchically such that a building type can be used to represent all buildings below it in the hierarchy. *Overhang generic building* is the most general, and *rectangular* and *peak roof buildings* are the most specific models. The more specific a building model is, the fewer parameters are needed to completely specify it. The required and optional parameters for each building type are defined in Section 4

2.2 Overview of Exchange Structure and Syntax

The structure of the CMU MAPSLab Site Exchange Format is defined in this section. A valid exchange file is partitioned into a collection of data blocks. The first data block must be the *file attribute block* and defines a set of data that describes the creation of this interchange file. The second data block must be the *world block*. It defines the local origin of the site model and number of building objects in the interchange file (see Section 3).

Each of the remaining data blocks are *object blocks* and each defines an individual object being exchanged. Valid *object blocks* are *building blocks*, *constraint blocks*, *surface blocks*, *road blocks* and *road intersection blocks*. The contents of each data block must appear in the order defined in this document. An exchange

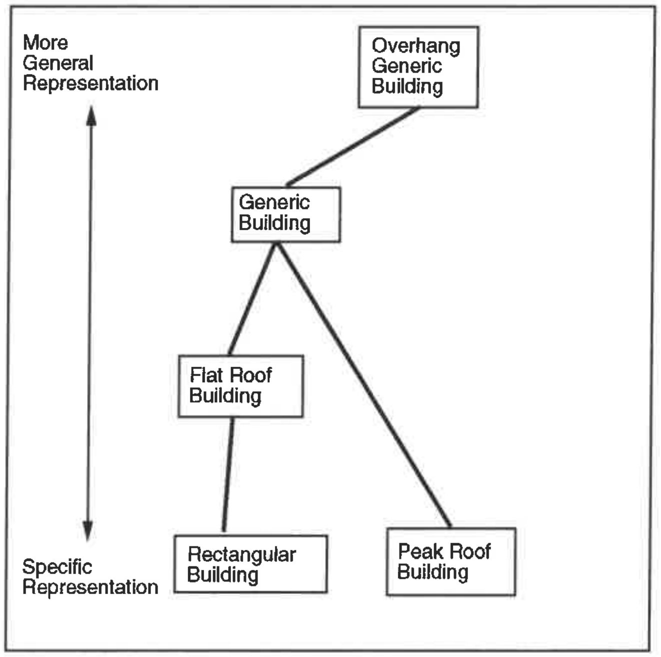

Figure 2: Hierarchical Building Classes

file will have no *object block* data blocks only if there are no objects represented in this world.

```
<Exchange File> :=
Begin File::
  <File Attribute Block>
  <World Block>
  <Building Block>|<Constraint Block>|<Surface Block>|<Road Block>|<Road Intersection Block>
  <Building Block>|<Constraint Block>|<Surface Block>|<Road Block>|<Road Intersection Block>
  ...
End File
```

The exchange file is delimited by the strings **Begin File::** and **End File**. The contents of any data block must be indented by two white spaces from the **Begin** delimiter.

2.3 Attribute Definition

Attributes are used to record optional and required information about the interchange file or properties of the imagery, sensor models, buildings, surfaces, roads, or intersections being exchanged. The CMU Site Exchange Format defines a set of required *file attributes* as described in Section 2.4. Each *object* and *world block* has an optional *attribute block* for defining user defined attributes at the world or object level. The syntax of the *attribute block* is defined below. Attributes consist of user defined information that lies outside the site model exchange definition but may be useful to transmit within the site model exchange file. Some plausible user defined attributes include the following properties:

```
road material: asphalt
building wall material: cinder block
building construction date: 8/20/85
```

As the site model interchange format evolves, some standard attributes may be defined and used in the *object* or *world block*.

In order to simplify parsing, the *attribute block* is not optional. If a *object block* has no attributes, the *attribute block* must still be present; however, the **Number of Attributes** field in the block will have value of 0.

```
<Attribute Block> :=
Begin attributes::
  Number of Attributes: <n>
  <attribute 1>: <value 1>
  <attribute 2>: <value 2>
  ...
  <attribute n>: <value n>
End attributes
```

The *attribute block* is delimited by **Begin attributes::** and **End attributes**. The first field in the attribute block, **Number of Attributes**, defines the number of attributes to follow. The attribute and value are separated by a colon. The attribute and value are strings and must not contain colons.

2.4 File Attribute Block

The *file attribute block* contains required data regarding the Site Exchange file. The purpose of these attributes is to make the file more expository and somewhat self documenting. However, none of these attributes are used or required for the interchange of site model data.

The *producer name* is intended to store the name of the program or system that produced the site model contained in the exchange file. Currently, CMU systems capable of writing these exchange files include SITECITY, VHBUILD and PIVOT. Using the CMU-SITE-API other programs, including traditional manual modeling systems, can identify the source of the site model data.

The *date* and *version* attributes should define when the exchange file was generated and the version of the CMU Site Exchange API that was used to produce this file.

The *title* field can be used to briefly describe the contents of the Site Exchange file.

```
<File Attribute Block> :=
Begin file attributes::
  Producer: <producer name>
  Date: <date value>
  Version: <version name>
  Title:   <title name>
End file attributes

<producer name> := <string>
<date value>    := <month>:<day>:<year>
<version name>  := CMU-Site-Exchange <version number>
<version number>:= <number>
<title name>    := <string>
```

3 World Definition

The world block defines the various coordinate systems required to rigorously locate the building site model descriptions, defined in a local coordinate system, to an absolute position on the earth.

Ellipsoid Name: defines the reference ellipsoid (geometric model of the earth) for this site model. Valid ellipsoid names include **WGS_1984**, **CLARKE_1866** and **BESSEL_1841**. All data should be relative to **WGS_1984**, unless there are compelling reasons to use another ellipsoid. This is current DMA policy for all new data. If the ellipsoid and/or datum are not commonly known ones, then the parameters should be given in the attribute block.

Horizontal Datum: and **Vertical Datum:** define the horizontal and vertical datums to which the data is referenced. Valid horizontal datums include **WGS_1984**, **BESSEL_1841** and **NAD27**, while vertical datums include **NAVD29**, **MSL** and **BESSEL_1841**. Again, all data should be relative to **WGS_1984** and **MSL**, unless there are compelling reasons otherwise. This is current DMA policy for all new data.

Local Origin: defines the origin of the local vertical coordinate system (Section 3.1). The origin is defined as a geodetic value. N and S define the north and south hemisphere for the latitude. E and W define the east and west facing for the longitude. The remaining values define the *degree*, *minute*, *second* and *thousandth of second* for the geodetic coordinate.

The elevation is expressed in units of meters.

Geocentric to Local Matrix: defines the rotation matrix which rotates the geocentric coordinates system into the local coordinate system. The geocentric coordinate system, (sometimes known as Earth-Center Fixed), has its origin at the center of the ellipsoid. The X axis lies in the plane of the equator and goes through the line of 0 longitude (Greenwich meridian), while the Z axis points through the North pole. The Y axis completes a right-handed coordinate system.

The next field is the attribute block, which defines the optional user defined attributes associated with the file. The syntax for the attribute block is defined in Section 2.3.

```
<World Block> :=
Begin world::
  Ellipsoid Name: <ellipsoid name>
  Horizontal Datum: <horizontal datum name>
  Vertical Datum: <vertical datum name>
  Local Origin: <geodetic value>
  Geocentric to Local Matrix: <matrix values>
  <Images Block>
  <Attribute Block>
  Number of Objects: <no>
End world

<ellipsoid name> := <string>
<geodetic value> :=
  <N|S> <deg> <min> <sec> <tho> <E|W> <deg> <min> <sec> <tho> <elevation>
<matrix values> :=
  <v11> <v12> <v13> <v21> <v22> <v23> <v31> <v32> <v33>
```

The definition for the world block is delimited by **Begin world::** and **End world** and is defined below. The last field is **Number of Objects:** which defines the number of object data blocks that follow in the interchange file.

3.1 Local vertical coordinate system

The local vertical coordinate system is a Cartesian system with its origin defined at an arbitrary location (X_0, Y_0, Z_0 in geocentric coordinates) on the earth's surface. The local coordinate system has units of meters. The X axis points east, the Y axis north, and the Z axis is vertical, pointing away from the earth's surface. The transformation from geocentric to local coordinates is based on the latitude (ϕ) and longitude (λ) of the origin ([7, p. 485]):

$$\begin{bmatrix} X_L \\ Y_L \\ Z_L \end{bmatrix} = M_{LG} \begin{bmatrix} X_G - X_0 \\ Y_G - Y_0 \\ Z_G - Z_0 \end{bmatrix} \tag{1}$$

$$M_{LG} = \begin{bmatrix} v11 & v12 & v13 \\ v21 & v22 & v23 \\ v31 & v32 & v33 \end{bmatrix} = \begin{bmatrix} -\sin\lambda & \cos\lambda & 0 \\ -\sin\phi\cos\lambda & -\sin\phi\sin\lambda & \cos\phi \\ \cos\phi\cos\lambda & \cos\phi\sin\lambda & \sin\phi \end{bmatrix} \tag{2}$$

3.2 Storing Image Information

The 3D position of a building model point is often measured in multiple images. In order to specify the correspondence between the observed position in one or more images and the calculated world position it is necessary to store the image measurements associated with each point. In the CMU Site Exchange format a list of images, used for site model measurement, must be defined as an **images block** within the **world block**. These images are referred to using a common name, such as **j1** or **m2** of the RADIUS model board image set. In addition, a standard interchange format for the camera models used to calculate the 3D world position of the model must be included. These requirements make it possible for users to interchange:

- Sets of points derived from manual, semi-automated, or fully automated procedures that correspond to semantic points in the building model, ie, *peak point*, *roof corner*, *floor corner*.
- The computed 3D world location for a set of corresponding points using a set of rigorous camera models. Thus users can experiment and verify these solutions using their own sensor model implementations.
- Allow for alternative methods for object model fitting and manipulation by going back to the original image measurements.
- Allow for incremental update of the 3D building models by providing image measurements in addition to the current solved 3D building position.

There are currently two defined formats: the TEC [4, 5] header file format or the FBIP interchange format [1]. The TEC or the FBIP files must also be included in the distribution of the site models. Sample files are included at the end of this document.

```
Begin images::
  Number of Images: <ni>
  Image 0: <image name 0>
  Header 0: <name of the TEC or FBIP header file for image 0>
  ...
  Image ni-1: <image name ni-1>
  Header ni-1: <name of the TEC or FBIP header file for image ni-1>
End images
```

4 Building Definition

A *building block* is used to store the topology and geometry of a building object in the interchange file, and is delimited by **Begin building model::** and **End building model**. The first field within the *building block* is the **Model Name:** field, which defines the name of the model. The model name must be a unique string identifier within each interchange file.

The *model parameter block* defines the building model type and parameters associated with the model. See Section 4.1. The *point list block* defines a list of vertices on the building object. See Section 4.2 and Section 4.3. Finally, the *attribute block* defines the optional attribute-value pairs associated with the building object.

```
<Building Block> :=
Begin building model::
  Model Name: <name>
  <Model Parameter Block>
  <Point List Block>
  <Attribute Block>
End building model

<Model Parameter Block> :=
  <Rectangular Flat Roof Parameters> |
  <Flat Roof Parameters> |
  <Peak Roof Parameters> |
  <Generic Roof Parameters> |
  <Overhang Generic Roof Parameters>
```

4.1 Model Parameter Block

Different building models require different sets of model parameters. The number of points on the building object and the semantics of the *point id* for these points also depend on the building model type. The *point id* is used to define the topological position of a point in the building object.

Currently, there are five different types of building models defined in the CMU MAPSLab Site Exchange Format. These building models can be organized hierarchically, as previously shown in Figure 2, such that a building type can represent all buildings below it in the hierarchy. *Overhang generic building* is the most general, and *rectangular* and *peak roof buildings* are the most specific models. The more specific a model is, the fewer parameters are needed to completely specify it. In the following sections the required parameters for each of the five building types supported in this interchange are specified.

4.1.1 Flat Roof Parameters

The *flat roof parameters block* defines a flat roof building object (Figure 3). A flat roof building object has a horizontal roof and floor connected by vertical walls. The floor and roof must have the same number of vertices. The number of vertices on the roof and floor must be greater than two.

```
<Flat Roof Parameters> :=
Begin flat roof parameters::
  Number of Floor Points: <nfpts>
  Floor Elevation: <elev>
  Model Height: <height>
End flat roof parameters
```

The *flat roof parameters block* is delimited by **Begin flat roof parameters::** and **End flat roof parameters**. The first field defines the number of floor points in the building object. Therefore, the number of points in the *point list block* for a *building block* with *flat roof parameters block* must be twice the number of floor points. The *point id* of the points within the **point list block** must be consecutive from 0 to $2nfpts - 1$. Points 0 to $nfpts - 1$ define the floor polygon in counter clockwise order in the local coordinate system. Points $nfpts$ to $2nfpts - 1$ define the roof polygon also in the counter clockwise order in the local coordinate system. Points i and $i + nfpts$ form a vertical edge.

The next two fields, **Floor Elevation:** and **Model Height:** define the parameters of the flat roof building for the idealized model given the position of the points in the *point list block*. Since the local coordinate of a point is triangulated using image measurements, the building might not be *geometrically correct* due to errors in the camera model. Therefore, these parameter values are used to define the *idealized* object. The value of **floor elevation:** defines the z value of the floor. The xy position of the floor vertices are taken from points 0 to $nfpts - 1$ of the **point list block**. The value of **model height** defines the distance between the roof polygon and the floor polygon.

$$model\ height = \frac{\sum_{i=0}^{nfpts-1} z(p_{i+nfpts}) - z(p_i)}{nfpts} \tag{3}$$

$$floor\ elevation = \frac{\sum_{i=0}^{nfpts-1} z(p_i)}{nfpts} \tag{4}$$

$z()$ returns the local z value of a point and p_i is the i-th point.

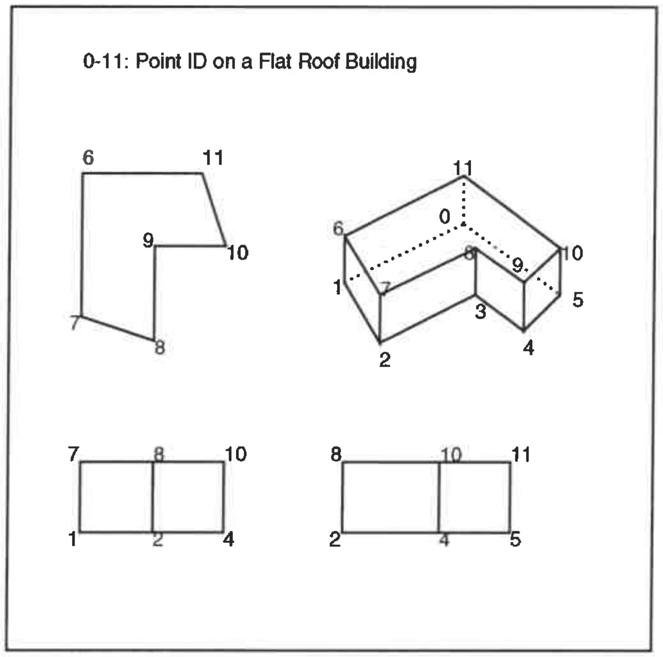

Figure 3: An example of flat roof building

4.1.2 Rectangular Flat Roof Parameters

The *rectangular flat roof parameters block* describes a subclass of flat roof building object whose roofs are rectangular (Figure 4).

```
<Rectangular Flat Roof Parameters> :=
Begin Rectangular Flat Roof Parameters::
  floor elevation: <elev>
  model height: <height>
  model length: <length>
  model width: <width>
End Rectangular Flat Roof Parameters
```

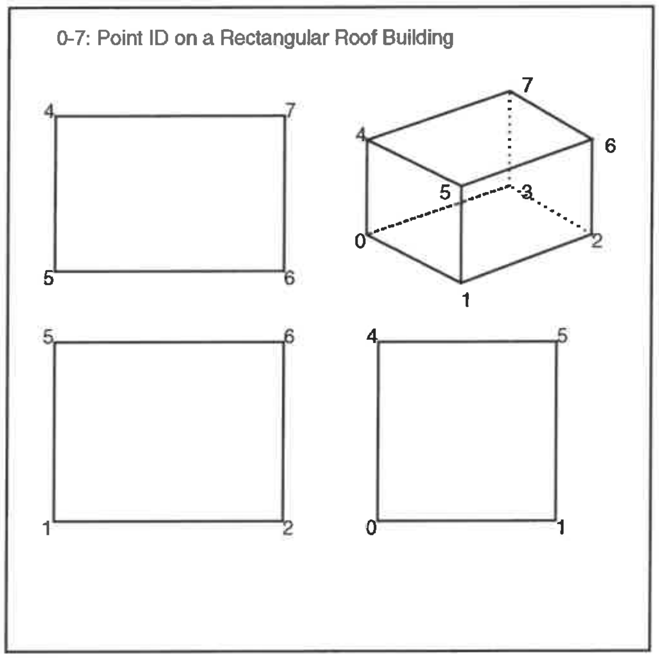

Figure 4: An example of rectangular flat roof building

There should be exactly 8 points in the point list. The parameters of height, length and width are defined by the following equations using the points in the point list.

$$model\ height = \frac{\sum_{i=0}^{3} z(p_{i+4}) - z(p_i)}{4} \tag{5}$$

$$model\ length = \frac{dist(p_0, p_1) + dist(p_2, p_3) + dist(p_4, p_5) + dist(p_6, p_7)}{4} \tag{6}$$

$$model\ width = \frac{dist(p_1, p_2) + dist(p_0, p_3) + dist(p_5, p_6) + dist(p_4, p_7)}{4} \tag{7}$$

$dist()$ returns the distance between two points.

4.1.3 Peak Roof Parameters

The *peak roof parameters block* defines a peak roof building object. The defined peak roof model is shown in Figure 5. The peak roof building object has two components, the rectangular flat roof building component and the peak component. A peak roof building must have 10 points in the *point list block*, with *point id* of the points in the point list numbering from 0 to 9 and having semantics as shown in Figure 5. Points 0 to 3 define the floor polygon in the counter-clockwise direction. Points 4 to 7 define the rectangular roof and points 8 and 9 define the peak edge.

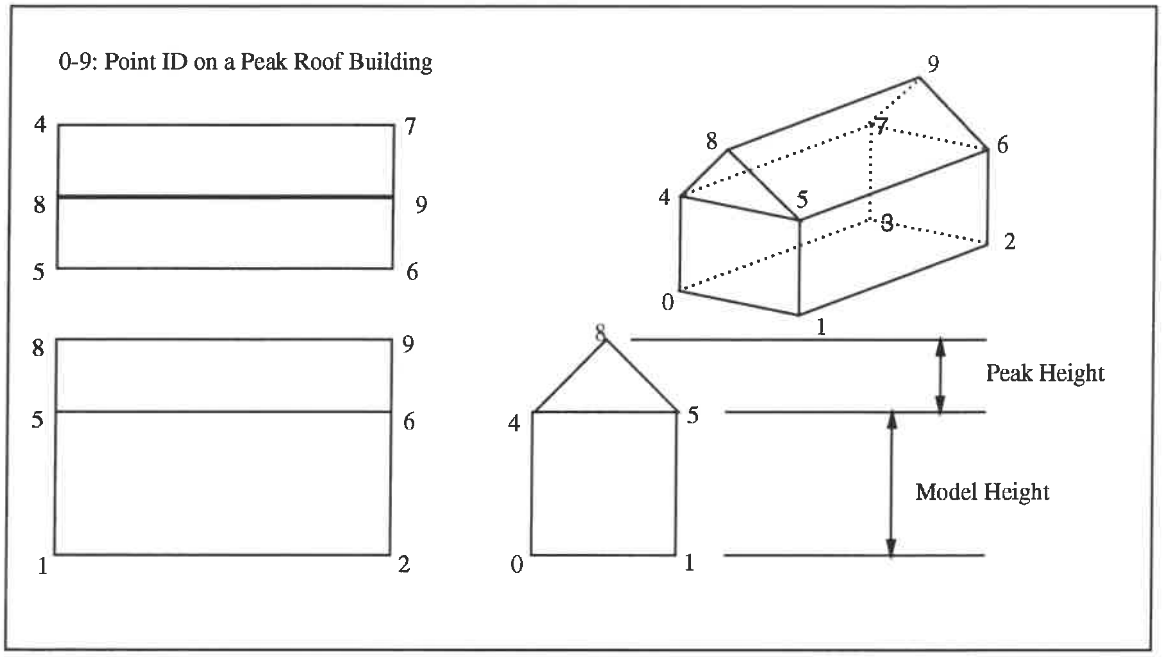

Figure 5: Peak Roof Building

```
<Peak Roof Parameters> :=
Begin peak roof parameters::
  Floor Elevation: <elev>
  Model Height: <height>
  Peak Height: <pheight>
End flat roof parameters
```

Like the flat roof building, the *peak roof parameters block* defines the parameters for the idealized peak roof building. The value of **Floor Elevation:** defines the z value of the floor. The xy position of the floor vertices are taken from point 0 to 3 of the **point list block**. The value of **Model Height:** defines the distance of the roof polygon away from the floor polygon, and **Peak Height:** defines the distance of the peak edge to the rectangular roof of the peak roof building as shown in Figure 5.

$$\textit{model height} = \frac{\sum_{i=0}^{3} z(p_{i+4} - z(p_i)}{4} \tag{8}$$

$$\textit{peak height} = \frac{z(p_8) + z(p_9)}{2} - \frac{\sum_{i=4}^{7} z(p_i)}{4} \tag{9}$$

$$\textit{floor elevation} = \frac{\sum_{i=0}^{3} z(p_i)}{4} \tag{10}$$

4.1.4 Generic Roof Parameters

The *generic roof parameters block* defines a building type that can have an arbitrary roof structure. The roof of a generic roof building is assumed to be composed of polygonal facets. Each roof facet should be planar. Since a generic roof building does not have a defined topology that can be specified using a set of parameters, the geometric structure of a generic roof building is defined by the points in the *point list block* and the topology of the roof structure is defined by the *roof polygon block*.

```
<Generic Roof Parameters> :=
Begin generic roof parameters::
  Number of Floor Points: <nfpts>
  Number of Roof Polygons: <nroofs>
  <Roof Polygon Block>
  ...
  <Roof Polygon Block>
End generic roof parameters

<Roof Polygon Block> :=
Begin roof polygon::
  Number of Roof Points: <nrpts>
  point 0: <p0>
  ...
  point <nrpts>-1: <p(nrpts-1)>
End roof polygon
```

There should be at least $2nfpts$ points in the *point list block*. The list of point IDs should be sequential starting from 0. The floor of the generic roof building is defined by points 0 to $nfpts - 1$ in the counter-clockwise direction in the local coordinate system. The exterior roof boundary of the generic roof building is defined by points *nfpts* to $2nfpts - 1$ also in the counter clockwise direction in the local coordinate system. Point i and $i + nfpts$ correspond to form a vertical edge.

Points with point ID values greater than $2nfpts - 1$ are the interior points of the roof structure and they are used by the *roof ID block* to define the topology of the roof structure. The value *nroofs* defines the number of polygonal facets on the roof structure. If it has a value of 0, then the generic roof building defines a building with one planar roof. There should be *nroofs Roof Polygon Blocks* following the *number of roof polygons* field in the *generic roof parameters block*.

Each *roof polygon block* defines a roof polygon on the roof structure. The value of **Number of Roof Points** defines the number of vertices on the roof polygon. This is followed by a list of point IDs corresponding to the point ID of the points in the *point list block*. The order of the point IDs should be in counter clockwise order in the local coordinate system. The point IDs used in the *roof polygon block* should consist of point IDs with value of *nfpts* and greater, because points whose point id value is less than *nfpts* represent floor vertices.

An example of the semantics of the point IDs and the roof polygons of a generic roof building is shown in Figure 6. There are a total of 9 points in the building structure. Points 0 to 3 define the floor polygon. Points 4 to 7 define the roof boundary corresponding to the floor polygon. There are a total of 4 roof polygons, with 3 points in each facet.

4.1.5 Overhang Generic Roof Parameters

The *overhang generic roof* model is similar to the generic roof model, except that the roof facets extend over the walls of the buildings. It requires more points to define the tops of the walls since these points no longer coincide with the boundary of the roof structure.

```
<Overhang Generic Roof Parameters> :=
Begin overhang generic roof parameters::
```

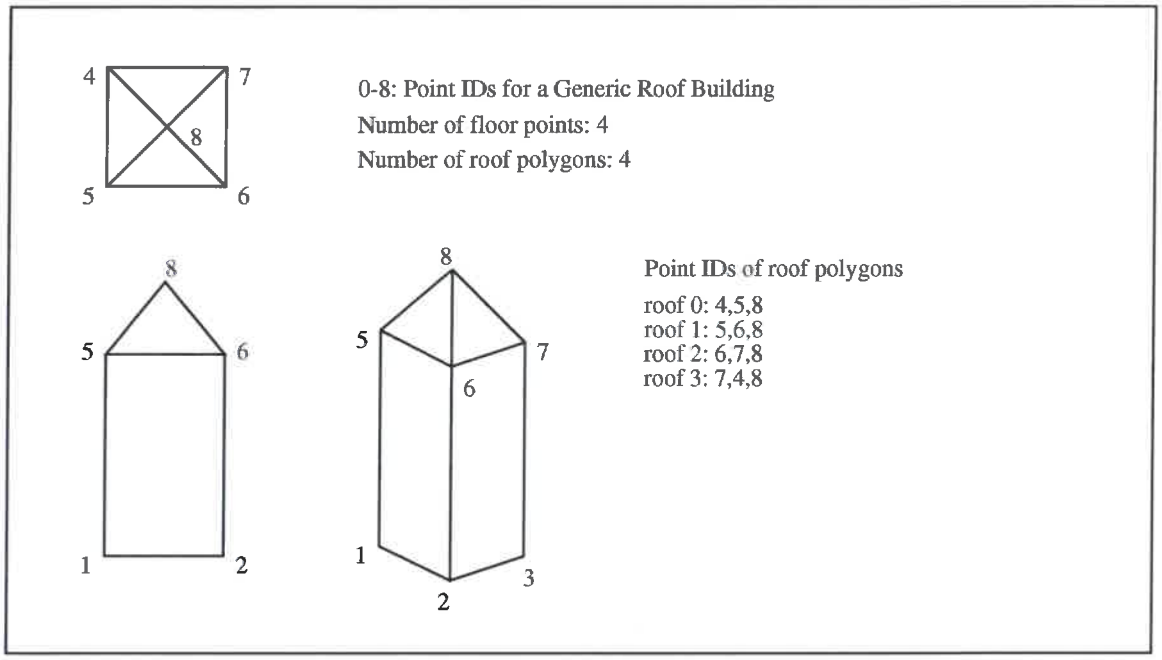
0-8: Point IDs for a Generic Roof Building
Number of floor points: 4
Number of roof polygons: 4
Point IDs of roof polygons
roof 0: 4,5,8
roof 1: 5,6,8
roof 2: 6,7,8
roof 3: 7,4,8
4
7
8
5
6
1
2
3

Figure 6: Generic Roof Building

```
  Number of Floor Points: <nfpts>
  Number of Roof Polygons: <nroofs>
  <Roof Polygon Block>
  ...
  <Roof Polygon Block>
End overhang generic roof parameters
```

There should be at least $3nfpts$ points in the *point list block*. The list of point IDs should be sequential starting from 0. The floor of the generic roof building is defined by points 0 to $nfpts - 1$ in the counter clockwise direction in the local coordinate system. Points *nfpts* to $2nfpts - 1$ define the extrusion of the floor to the roof level to form the walls of the building model. The exterior roof boundary of the generic roof building is defined by points $2nfpts$ to $3nfpts - 1$ also in the counter clockwise direction in the local coordinate system. Points i and $i + nfpts$ correspond to form a vertical edge, Points i, $i + nfpts$ and $i + 2nfpts$ are considered to be corresponding vertices. However, point $i + 2nfpts$ does not need to be topologically connected with the other two points.

The **roof polygon block** is the same as those defined for the **generic roof parameters**. The point IDs used in the **roof polygon block** should consist of point IDs with value of $2nfpts$ and greater.

An example of the semantics of the point IDs and the roof polygons of an overhang generic roof building is shown in Figure 7. There are a total of 13 points in the building structure. Points 0 to 3 define the floor polygon. Points 4 to 7 define the top of the walls and are usually not visible. Points 8 to 11 define the roof boundary. There are a total of 4 roof polygons, with 3 points in each facet.

4.2 Point List Block

The *point list block* is used to define a list of points that defines the building object and has the following form:

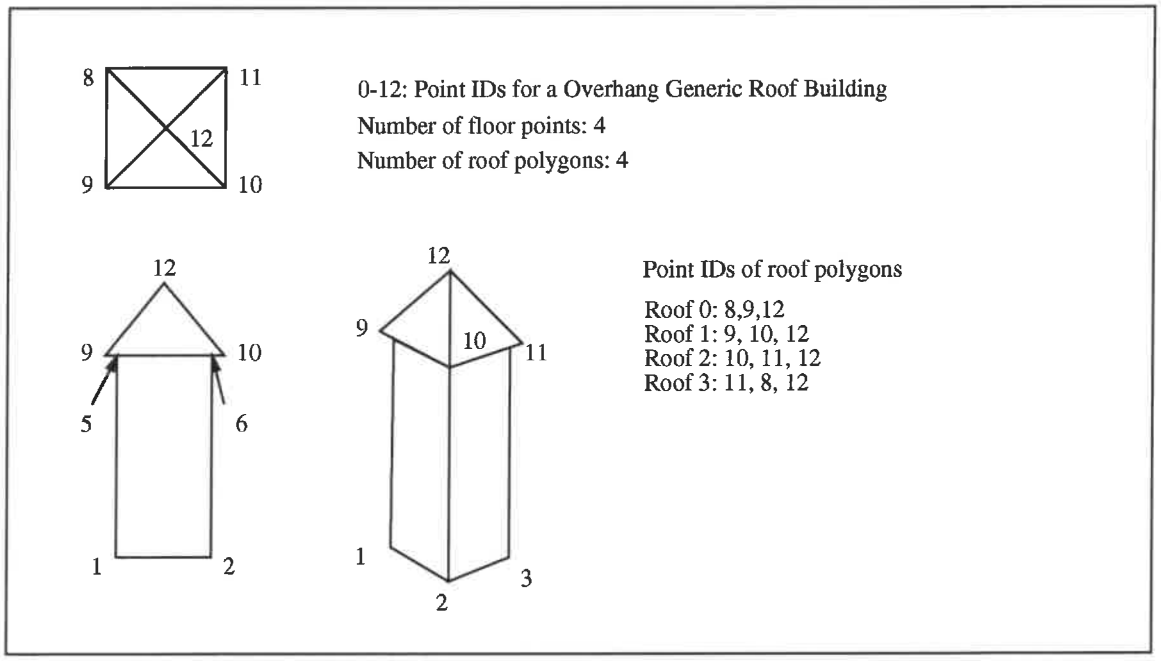
8
11
12
9
10
0-12: Point IDs for a Overhang Generic Roof Building
Number of floor points: 4
Number of roof polygons: 4
12
9
10
5
6
1
2
12
9
10
11
1
2
3
Point IDs of roof polygons
Roof 0: 8,9,12
Roof 1: 9, 10, 12
Roof 2: 10, 11, 12
Roof 3: 11, 8, 12

Figure 7: Overhang Generic Roof Building

```
<Point List Block> :=
Begin pointlist::
  Number of Points: <n>
  <Point Block>
  ...
  <Point Block>
End pointlist
```

The *point list block* is delimited by **Begin pointlist::** and **End pointlist**. The first field must be **Number of Points:**, which defines the number of *point block* to follow. The fields within the *point list block* are indented by two white spaces.

4.3 Point Block

The *point block* is used to define the position and covariance of a point in the local coordinate system defined in the world block, along with the image position of the point in the images. The point block can occur only within the *point list block*.

```
<Point Block> :=
Begin point::
  Point ID: <id>
  Local Coordinate: <x> <y> <z>
  Local Covariance: <uxx> <uyy> <uzz> <uxy> <uyz> <uxz>
  Number of Image Measurements: <npi>
  Image 0: <r0> <c0> <sigma0>
  ...
  Image <npi>-1: <r(npi-1)> <c(npi-1)> <sigma(npi-1)>
End point
```

The *point block* is delimited by the **Begin point::** and **End point**. The first field in the *point block* must be the **Point Id:** field which defines an identifier for the point. The value of *point id* must be an integer and unique within the scope of the **point list block**. The purpose of the point id is to define the location of a point within a building model as shown in Section 4.1. The **Local Coordinate** of a point is defined with respect to the local coordinate system defined in the *world block*. The values x, y and z have units of meters. Likewise, the **Local Covariance** values are also expressed with respect to the local coordinate system defined in the *world block*. A point that does not have covariance information is assumed to be perfectly positioned and should have values of 0 in the **Local Covariance** field. The values of **local covariance** define the covariance matrix:

$$L_{cov} = \begin{bmatrix} uxx & uxy & uxz \\ uxy & uyy & uyz \\ uxz & uyz & uzz \end{bmatrix} \tag{11}$$

Number of Image Measurements: is a required field after the **Local Covariance:** field. The value *npi* defines the number of image measurements used to estimate the 3D position of the point. If the value of *npi* is zero, then the **End point** terminates the *point block*. Otherwise, the next *npi* lines define the image measurements and the uncertainties of the image measurements and has the following form:

```
Image <i>: <row> <column> <sigma>
```

Image i refers to the i^{th} image defined in the **world block**(Section 3.2). *row* and *column* are the image measurements of the point in units of pixels, in **Image i**, and they are expressed as floating point values in the image coordinate system previously described in Figure 1. *sigma* is the uncertainty of the image measurement in pixels, and is expressed as a floating point value. The value of *sigma* should reflect the expected accuracy of the image measurement. If an image measurement is treated as a Gaussian distribution function uniform in the *rows* and *columns*, to account for measurement error, then the *row* and *column* values are the average or the expected position of the Gaussian function. The *sigma* represents the standard deviation of the distribution.

5 Constraints

We will occasionally encounter a complex building that cannot be represented using an existing building type, defined in section 3, but can be constructed using the existing building types as primitives. One problem of using the multiple buildings to construct a complex building object is that the composite 3D building object often does not agree with our expectation. The composite object can have gaps between walls or intersects, and in general, just does not look right. Therefore, we use geometric constraints to sew these primitive objects together to form a seamless 3D building.

5.1 Constraint Types

There are currently three main types of constraints: coplanar, collinear, and angle:

- **Coplanar:** All points on the selected objects form a plane. The equation for the plane is $Ax + By + Cz + D = 0$. A, B, C and D are the parameters for the coplanar constraint.
- **Collinear:** All points on the selected objects forms a line. The equation for the line is $\frac{x-X0}{A} = \frac{y-Y0}{B} = \frac{z-Z0}{C}$. $A, X0, Y, B0, Z$ and $C0$ are the parameters.
- **Angle:** Three points on the same structure form an angle, with the point in the middle being the vertex of the angle. All selected points must be on the same structure, since the connecting line segments between points are used to determine which point should be the vertex. The angle is measured in radians.

5.2 Constraint Definition

A *Constraint Block* is used to store object constraints in the interchange file. It is delimited by **Begin constraint::** and **End constraint**. The first field within the *constraint block* is the **name:** field, which defines the name of the constraint. The name field value must be a unique string identifier within the interchange file.

The second field within the constraint block is the type field, **type:**. This field defines the type of constraint being defined, such as coplanar or collinear. Valid values for this field are **COPLANAR, COLLINEAR, and ANGLE** and are defined in Section 5.1.

The parameters field, **params:**, lists the parameter coefficients for this constraint, and are dependent on the constraint type. These parameters are described in Section 5.1

The next field, **npts:**, specifies the number of points that this constraint is applied to.

The number of points field is followed by the point list. Each point in the list consists of 2 field values: the point ID and the location. The point ID value is the unique identifier of an object in the interchange file. This is used to find the objects being constrained in the interchange file, as all objects are identified by a unique string. The location field value, ¡loc¿, indicates which point in that object is actually being constrained. The purpose of this field is to identify the location of the point being constrained within the object model, as shown in Section 4.1.

The final field is an attribute block, which is defines the optional attribute-value pairs associated with the constraint model.

```
<Constraint Block> :=
Begin constraint::
  name: <id>
  type: <type>
  params: <str>
  npts: <n>
  pt 0: <obj 0> <loc>
  pt 1: <obj 1> <loc>
  ...
  pt n-1: <obj n-1> <loc>
  <Attribute Block>
End constraint
```

In Figure 8, a complex building is created from 2 rectilinear flat roof buildings using coplanar constraints. The floors of the 2 buildings are constrained to be coplanar, as are the adjoining walls of the buildings.

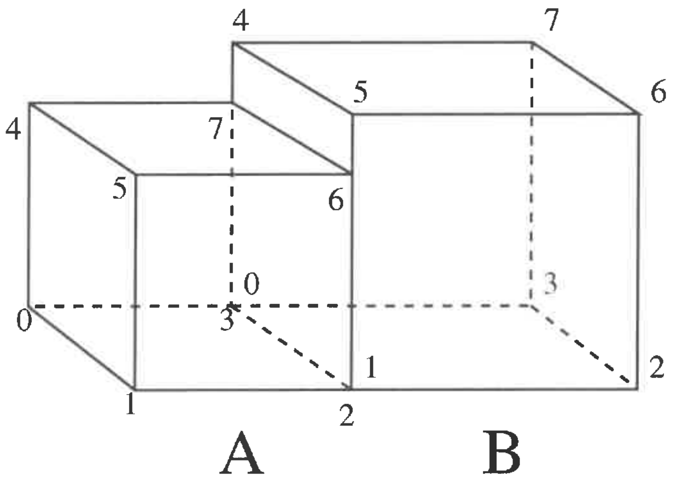

Figure 8: Complex building from 2 rectilinear flat roof buildings

6 Surface Definition

A *Surface Block* is used to store the topology and geometry of a surface object in the interchange file. It is delimited by **Begin surface::** and **End surface**. The first field within the surface block is the **name:** field, which defines the name of the surface model with a unique identifier.

The **material:** identifier defines the type of surface that is being defined. The **function:** identifier gives a more specific description of the type of surface being identified.

These fields are followed by a *Point List Block*. This defines the list of vertices on the surface object. See section 3.2 and 3.3 for a description of the point list block and point block.

The final field is an attribute block, which is defines the optional attribute-value pairs associated with the surface model.

```
<Surface Block> :=
  Begin surface::
    name: id
    material: <str>
    function: <str>
    <Point List Block>
    <Attribute Block>
  End surface
```

The surface example in Figure 9 defines an asphalt parking lot.

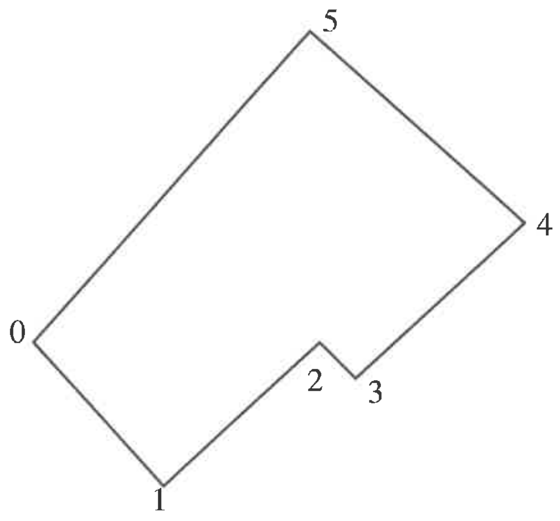

Figure 9: Surface

7 Road Definition

The *Road Block* is used to store the topology and geometry of a road object in the interchange file, and is delimited by **Begin road::** and **End road**. The first field within the road block is the **name:** field, which defines the name of the road model with a unique identifier.

This is followed by the number of points field, **npts:**, which defines the number of points that make up the centerline of the road in the interchange file. Following the number of points field, is the list of road points, with each road point defined by a *Road Point Block*.

```
<Road Block> :=
    Begin road::
      name: <id>
      npts: <n>
      <Road Point Block>
      ...
      <Attribute Block>
    End road
```

7.1 Road Point Block

The *Road Point Block* is delimited by **Begin road point::** and **End road point**. It consists of a *Point Block*, defined in Section 4.3, and a **width:** field. The **width:** field defines the width of the road at that point.

```
<Road Point Block> :=
  Begin road point::
    name: <id>
    <Point Block>
    width: <w>
  End road point
```

Note that road points A2 and B1 are not the same point. They are two separate points with different *name:*s. They can be grouped together as an intersection using the *Road Intersection Block*.

7.2 Road Intersection Block

The *Road Intersection Block* is used to group *Road Points* into intersections. The first field in the *Road Intersection Block* is the **name:** field, which defines the name of the road intersection with a unique identifier. This is followed by a *Point Block* which defines the location of the intersection. The next field is **npts:** which defines the number of road points included in the intersection. The list of intersection points contains two fields: the road name and the location. The road name is the unique identifier of the road that the point is a member of, and the location indicates the order of that point in the road. The last element in the *Road Intersection Block* is an *Attribute Block*

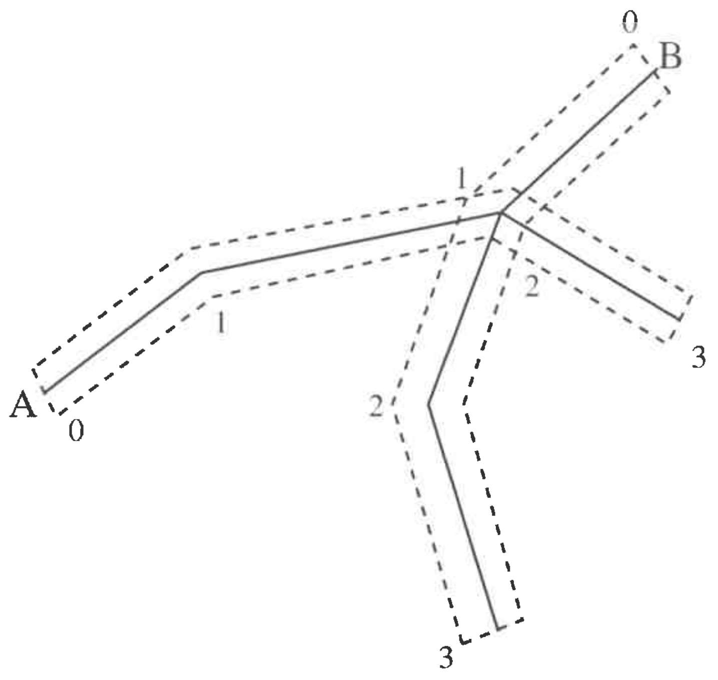

Figure 10: Intersecting Roads

```
<Road Intersection Block> :=
  Begin road intersection::
    name: <id>
    <Point Block>
    npts: <num>
    Begin road intersection points::
      pt 0: <road_name 0> <loc_0>
      ...
      pt n: <road_name n> <loc_n>
    End road intersection points
    <Attribute Block>
  End road intersection
```

8 Description of an API for CMU MAPSLab Site Exchange Format

This section describes an Application Programmer Interface (API) for the site exchange format. The API consists of a set of data structures corresponding to the site exchange format and functions to read and write site exchange format files. The names of the data structures and functions are prefixed by **sef_**. The API is written in C and consists of 4 files, *sitexg-api.h*, *sitexg-api.c*, *sitexg-apiadt.c*, and *sitexg-coord.c*. A description of these functions are presented in the next two subsections in the UNIX manual page format.

Following the API description are three sample programs illustrating the use of the API. The first program gives an example of a simple C language program to create a site model using the CMU MAPSLab Site Exchange API primitives. The site model contains one rectangular flatroof building object that is a unit cube.

The second program gives an example of a simple C language program constructed using the API primitives that reads in a CMU MAPSLAB Site Exchange Format file and prints the number of buildings, surfaces, roads and constraints, as well as information about the objects. The program then translates all the buildings with a directional vector of [10.0,10.0,10.0] meters in the local coordinate system, while leaving the local covariance and image measurements unaffected.

The third program is similar to the first program, but the object being produced is a surface.

Finally, Section 8.6 provides a brief discussion of the steps necessary to develop a writer or reader for the Site Exchange Format using the API.

8.1 Sitexg-Api.5 Man Page

NAME

sitexg-api data structure for the site exchange api.

SYNOPSIS

#include <sitexg-api.h>

DESCRIPTION

This man page describes the data structure for the API for the site exchange format. These data structures correspond to the *block* structures of the exchange format.

Attribute Block

The structure of the attribute block is two arrays of strings:

```
typedef struct {
    int natt;
    char **attributes;
    char **values;
} sef_attributes;
```

World Block

The world block consists of the *ellipsoid* name, vertical and horizontal datum, the origin of the local coordinate system in term of geodetic values, latitude, longitude and elevation. The values of *v1* to *v9* are the geocentric to local coordinate transform matrix and is derived from the origin of the local system.
The values of *geo_x, geo_y,* and *geo_z* are the geocentric coordinates of the local coordinate system origin.
Valid ellipsoid names are: "WGS_1984", "BESSEL_1841", "CLARKE_1866"
Valid horizontal datum are: "WGS_1984", "NAD27", "BESSEL_1841"
Valid vertical datum are: "NAVD29", "MSL", "BESSEL_1841"
nobjs specifies the number of objects within the world and should not be manually modified.
images stores the image information as an attribute table. and *attributes* stores the attributes related to the site.

```
typedef struct {
    char *ellipsoid;
    char *vertical_datum;
    char *horizontal_datum;
    char lat_sgn, lon_sgn; /* the hemisphere designation */
    int lat_deg, lat_min, lat_sec, lat_thou;
    int lon_deg, lon_min, lon_sec, lon_thou;
    double elevation;
    double v1, v2, v3, v4, v5, v6, v7, v8, v9;
    double geo_x, geo_y, geo_z;
    int nobjs;
    sef_attributes *images;
    sef_attributes *attributes;
} sef_world;
```

Point Block

The point structure consists of an *id*, which gives the position of the point within a building model, its local coordinates, and covariances. In addition, the information about the image measurements related to this point is also stored in an array of the sef_imgpt structure.

```
typedef struct {
        int imgid;
        double r, c, sigma;
} sef_imgpt;

typedef struct {
        int id;
        double x, y, z;  /* local coordinate */
        double uxx, uyy, uzz, uxy, uyz, uxz; /* local covariance */
        int nimgs;
        sef_imgpt *imgpts;
} sef_pt;
```

Point List Block

This is simply a list of points.

```
typedef struct {
        int npts;
        sef_pt **pts;
} sef_ptlist;
```

Model Parameter Block

A structure is used for each type of building.

```
typedef struct {
        int nfpts; /* number of floor points */
        double floor_elev, model_height;
} sef_flatroof;

typedef struct {
        double floor_elev, model_height, model_length, model_width;
} sef_rect;

typedef struct {
        double floor_elev, model_height, peak_height;
} sef_peakroof;

typedef struct {
        int nfpts;
        int nroofs;
        int *roofpts;       /* array of number of points within each roof polygon */
        int **roofptids;   /* 2D array, value of [i][j] = the id of jth point in
                        * ith roof polygon */
} sef_genroof;
```

Building Block

This structure defines a building block. The value of *type* gives the type of the building, and the valid values are

```
#define SEF_FLATROOF 1
#define SEF_PEAKROOF 2
#define SEF_RECT 3
#define SEF_GENERIC 4
#define SEF_OVERHANG_GENERIC 5

typedef struct {
        char *name;
        sef_ptlist *ptlist;
        int type;
        union {
                sef_flatroof *flat;
                sef_peakroof *peak;
                sef_rect *rect;
                sef_genroof *genroof;
        } parameters;
        sef_attributes *attributes;
} sef_building;
```

File Attribute Block

The file attribute block contains information about the site models. All values are strings.

```
typedef struct {
        char *producer;
        char *date;
        char *version;
        char *title;
} sef_fileatt;
```

Surface Block

This structure defines a surface block.

```
typedef struct {
        char name[1024];
        sef_ptlist *ptlist;
        char mat[1024];
        char func[1024];
        sef_attributes *attributes;
} sef_surface;
```

Constraint Block

This structure defines a constraint block.

```
typedef struct {
        char name[1024];
        char inter[1024];
        char attr[1024];
        sef_attributes *attributes;
        int npts;
        char **id;
        int *order;
        void **pt;
} sef_constraint;
```

Road Block

This structure defines a road block.

```
typedef struct {
        char name[1024];
        int npts;
        sef_road_pt **roadpt;
        sef_attributes *attributes;
} sef_road;
```

Road Point Block

This structure defines a road point block. It contains a standard point block and an additional width field to define the road width.

```
typedef struct {
        char name[1024];
        sef_pt *pt;
        double width;
} sef_road_pt;
```

The Site

Finally, this structure stores world and building information about the site. nbuilds, nroads, ncons and nsurfs store the number of each type of object and should not be modified manually.

```
typedef struct {
        sef_fileatt fileatt;
        sef_world world;
        sef_building **buildings;
        sef_constraint **constraints;
        sef_road **roads;
        sef_surface **surfaces;
        int nbuilds;
        int nroads;
        int ncons;
        int nsurfs;
} sef_site;
```

8.2 Sitexg-Api.3 Man Page

NAME

sitexg-api – functions for the site exchange api.

SYNOPSIS

#include <sitexg-api.h>
link with -lm.

DESCRIPTION

This man page describes the manipulation functions for the API for the site exchange format. Unless otherwise specified, all functions return SEF_SUCCEED on successful completion, and SEF_FAILED on failure. All functions will have a sef_ prefix.

Creation

The following functions create new structures.
sef_site *sef_create()
creates and returns storage space for a site object.

sef_pt *sef_pt_create()
creates and returns storage space for a point object.

sef_building *sef_rect_create(char *name)
creates and returns storage space for a rectangular building with the given name.

sef_building *sef_flatroof_create(char *name)
creates and returns storage space for a flatroof building with the given name.

sef_building *sef_peakroof_create(char *name)
creates and returns storage space for a peakroof building with the given name.

sef_building *sef_genroof_create(char *name)
creates and returns storage space for a generic roof building with the given name.

sef_building *sef_overhang_genroof_create(char *name)
creates and returns storage space a overhang generic roof building with the given name.

sef_surface *sef_surface_create(char *name)
creates and returns storage space for a surface with the given name.

sef_constraint *sef_constraint_create(char *name)
creates and returns storage space for a constraint with the given name.

sef_road *sef_road_create(char *name)
creates and returns storage space for a road with the given name.

sef_road_pt *sef_road_pt_create(char *name)
creates and returns storage space for a road point with the given name.

Adding File Attributes

int sef_add_producer(sef_site *site, char *producer)
adds the producer string to the site.

int sef_add_date(sef_site *site, int month, int date, int year)
adds the date information to the site. Valid month values are between 1-12. Valid date values are between 1-31. It does not check for consistencies between month and date value.

int sef_add_date_string(sef_site *site, char *date)
adds the date information to the site.

int sef_add_title(sef_site *site, char *title)
adds the title string to the site.

int sef_add_version(sef_site *site, char *version)
adds the version string to the site.

Deletion

int sef_free(sef_site *site)
frees the contents of a entire site object.

Input/Output

int sef_write(FILE *f, sef_site *site)
writes a sef_site structure to a file in the site exchange format.

sef_site *sef_read(FILE *f)
reads a site exchange format file into the sef_site structure.

Adding Information to a World Block

int sef_add_world_ellipsoid(sef_site *site, char *ellipsoid_name)
adds the ellipsoid_name to the world block. The current value of ellipsoid name, if it exists, is deleted.

int sef_add_world_vertical_datum(sef_site *site, char *vertical_datum)
adds the vertical_datum to the world block. The current value of vertical_datum, if it exists, is deleted.

int sef_add_world_horizontal_datum(sef_site *site, char *horizontal_datum)
adds the horizontal_datum to the world block. The current value of horizontal_datum, if it exists, is deleted.

int sef_add_world_local_coord_lat(sef_site *site, char sgn, int deg, int min, int sec, int tho)
fills in the latitude coordinate of the origin of the local system.

int sef_add_world_local_coord_lon(sef_site *site, char sgn, int deg, int min, int sec, int tho)
fills in the longitude coordinate of the origin of the local system.

int sef_add_world_local_coord_elevation(sef_site *site, double elevation)
fills in the elevation of the origin of the local system.

int sef_add_world_image(sef_site *site, char *imgname, char *header)
adds the image name and Tec/Fbip header file name to the world block.

The geocentric to local coordinate system transformation matrix is computed automatically when the sef_site is written to a file. It should not be modified to ensure consistency with the defined origin of the local coordinate system.
int sef_add_world_attributes(sef_site *site, char *attribute, char *value)
adds a pair of attribute and value strings to the world block.

int sef_add_building(sef_site *site, sef_building *bld)
adds a building object to the site.

Coordinate Conversion and Origin Definition Support

int sef_add_world_local_coord_utm(sef_site *site, double x, double y, int zone, double z)
defines a local origin for the site file, placed at the given UTM coordinates. Can be used in place of sef_add_world_local_coord_lat,lon,elevation to set the origin. Assumes WGS1984 datum.

int sef_add_world_local_coord_geodetic(sef_site *site, double lat, double lon, double elev)
defines a local origin for the site file, placed at the given elevation and latitude/longitude (in radians). Can be used in place of sef_add_world_local_coord_lat,lon,elevation to set the origin. Assumes WGS1984 datum.

int sef_init_local_conversion(sef_site *site)
initializes the geocentric coordinates of the local origin in the site. An origin must be set in the site before using this routine. Assumes WGS1984 datum.

int sef_utm_to_geodetic(double utm_x, double utm_y, int zone, double lat, double lon)
converts UTM coordinates to latitude/longitude (in radians). Assumes WGS1984 datum.

int sef_geodetic_to_geocentric(double lat, double lon, double elev, double *x, double *y, double *z)
converts latitude/longitude (in radians) and elevation (in meters) to geocentric coordinates (in meters). Assumes WGS1984 datum.

int sef_geocentric_to_local(sef_site *site, double x, double y, double z, double *lx, double *ly, double *lz)
converts geocentric coordinates (in meters) to local coordinates (in meters). An origin must be set in the site and sef_init_local_conversion() must be called before using this routine. Assumes WGS1984 datum.

int sef_local_to_geocentric(sef_site *site, double lx, double ly, double lz, double *x, double *y, double *z)
converts local coordinates (in meters) to geocentric coordinates (in meters). An origin must be set in the site and sef_init_local_conversion() must be called before using this routine. Assumes WGS1984 datum.

int sef_geocentric_to_geodetic(double x, double y, double z, double *lat, double *lon, double *elev)
converts geocentric coordinates (in meters) to latitude/longitude (in radians) and elevation (in me-

ters). Assumes WGS1984 datum.

int sef_geodetic_to_utm(double lat, double lon, int desired_zone, double *utm_x, double *utm_y, int *zone)
converts latitude/longitude (in radians) to UTM coordinates. The desired_zone specifies a desired zone for the output UTM coordinates; this can be set to 0 for the routine to pick the closest zone. Assumes WGS1984 datum.

Adding Information to a Point Block

int sef_add_pt_id(sef_pt *pt, int id)
adds the point id to a point.

int sef_add_pt_xyz(sef_pt *pt, double x, double y, double z)
fills in the x-y-z local coordinate values of a point.

int sef_add_pt_covariance(sef_pt *pt, double uxx, double uyy, double uzz, double uxy, double uyz, double uxz)
fills in the local covariance values of a point

int sef_add_pt_imagept(sef_pt *pt, int imgid, double r, double c, double sigma)
adds values of an image measurement to a point.

Adding Information to a Building Block

int sef_add_building_attributes(sef_building *bld, char *att, char *val)
adds a pair of attribute and value strings to the building block.

int sef_add_building_point(sef_building *bld, sef_pt *pt)
adds a point block to the point list block of a building block.

int sef_add_building_num_floorpts(sef_building *bld, int nfpts)
int sef_add_building_peak_height(sef_building *bld, double height)
int sef_add_building_model_height(sef_building *bld, double height)
int sef_add_building_model_length(sef_building *bld, double length)
int sef_add_building_model_width(sef_building *bld, double width)
int sef_add_building_floor_elevation(sef_building *bld, double elev)
adds the defined parameter to the building block. If the building type does not allow this parameter, the value is not added and SEF_FAILED is returned.

int sef_add_building_roofptids(sef_building *bld, int npts, int *pts)
adds a list of point ids describing the roof polygon of a generic or overhang generic type building to the building block. If the building type does not allow the roof polygon description, the values are not added and SEF_FAILED is returned.

Adding Information to a Surface Block

int sef_add_surface_name(sef_surface *s, char *name)
int sef_add_surface_material_type(sef_surface *s, char *name)
int sef_add_surface_function_type(sef_surface *s, char *name)
int sef_add_surface_point(sef_surface *s, sef_pt *pt)
int sef_add_surface_attributes(sef_surface *s, char *att, char *val)

Adding Information to a Constraint Block

int sef_add_constraint_name(sef_constraint *c, char *name)
int sef_add_constraint_type(sef_constraint *c, char *type)
int sef_add_constraint_attributes_linear(sef_constraint *c, double A, double B, double C, double X0, double Y0, double Z0)
int sef_add_constraint_attributes_planar(sef_constraint *c, double A, double B, double C, double D)
int sef_add_constraint_attributes_angle(sef_constraint *c, double angle)
int sef_add_constraint_attributes(sef_constraint *c, char *att, char *val)

Adding Information to a Road Block

int sef_add_road_name(sef_road *r, char *name)
int sef_add_road_pt(sef_road *r, sef_road_pt *pt)
int sef_add_road_attributes(sef_road *r, char *att, char *val)

Adding Information to a Road Point Block

int sef_add_road_pt_name(sef_road_pt *pt, char *name)
int sef_add_road_pt_point(sef_road_pt *pt, sef_pt *sefpt)
int sef_add_road_pt_width(sef_road_pt *pt, double width)

Getting Information From File Attributes Block

char *sef_get_producer(sef_site *site)
gets the producer string from the site. returns 0 if failed.

char *sef_get_date(sef_site *site)
gets the date string from the site. returns 0 if failed.

char *sef_get_title(sef_site *site)
gets the title string from the site. returns 0 if failed.

char *sef_get_version(sef_site *site)
gets the version string from the site. returns 0 if failed.

Getting Information from a World Block

int sef_get_world_nobjs(sef_site *site)
returns the number of objects in the world. It returns -1 if it failed.

char *sef_get_world_ellipsoid(sef_site *site)
returns the ellipsoid name of the world block. It returns 0 on failure.

char *sef_get_world_horizontal_datum(sef_site *site)
returns the horizontal datum of the world block. It returns 0 on failure.

char *sef_get_world_vertical_datum(sef_site *site)
returns the vertical datum of the world block. It returns 0 on failure.

int sef_get_world_local_coord_lat(sef_site *site, char *hemi, int *deg, int *min, int *sec, int *thou)
int sef_get_world_local_coord_lon(sef_site *site, char *hemi, int *deg, int *min, int *sec, int *thou)
int sef_get_world_local_coord_elevation(sef_site *site, double *elevation)
gets the origin of the local coordinate system in the world block and store them in the arguments.

int sef_get_world_geocentric_to_local_matrix(sef_site *site, double *v1, double *v2, double *v3, double *v4, double *v5, double *v6, double *v7, double *v8, double *v9)
gets the values in the geocentric to local coordinate system matrix.

char *sef_get_world_image_name(sef_site *site, int i)
returns the name of the ith image definition in the images block of a world block. 0 is returned if the ith image is not defined.

int sef_get_world_image_id(sef_site *site, char *imgname)
returns the image id number given a image name from the world block in the site. -1 is returned if the imgname is not a valid image name in the world block.

sef_building *sef_get_building(sef_site *site, int i)
returns the ith building in the site file. 0 is returned on failure.

int sef_get_world_num_images(sef_site *site)
gets the number of images in the world block. It returns -1 on failure.

int sef_get_world_num_attributes(sef_site *site)
gets the number of attribute-value pairs in the world block. It returns -1 on failure.

char *sef_get_world_attribute(sef_site *site, int i)

char *sef_get_world_value(sef_site *site, int i)
get the ith attribute and value string from world block's attribute table. It returns 0 on failure.

Getting Information from a Point Block

int sef_get_pt_id(sef_pt *pt, int *id)
get the point id of a point block in the pointer .I id. This function returns SEF_SUCCEED if successful, SEF_FAILED otherwise.

int sef_get_pt_xyz(sef_pt *pt, double *x, double *y, double *z)
gets the values of the local coordinate of a point block.

int sef_get_pt_covariance(sef_pt *pt, double *uxx, double *uyy, double *uzz, double *uxy, double *uyz, double *uxz)
gets the values of the local covariance of a point block.

int sef_get_num_images(sef_pt *pt)
gets the number of image measurements in a point. It returns -1 on error.

int sef_get_pt_imagept(sef_pt *pt, int i, int *imgid, double *r, double *c, double *sigma)
gets the ith image measurement in the point block.

Getting Information from a Building Block

sef_pt *sef_get_building_point(sef_building *bld, int i)
gets the point block in the pointlist of a building block whose point id is *i*. It returns 0 if the point cannot be found.

int sef_get_building_num_points(sef_building *bld)
returns the number of points in the point list block of a building block.

int sef_get_building_floor_elevation(sef_building *bld, double *v)
int sef_get_building_num_floorpts(sef_building *bld, int *v)
int sef_get_building_num_roofs(sef_building *bld, int *v)
int sef_get_building_model_height(sef_building *bld, double *v)
int sef_get_building_model_length(sef_building *bld, double *v)
int sef_get_building_model_width(sef_building *bld, double *v)
gets the defined parameter of the building block and store in argument .I v. If the building type does not allow this parameter SEF_FAILED is returned, otherwise, SEF_SUCCEED is returned.

int *sef_get_building_roofptids(sef_building *bld, int i, int *npts)
gets the ith roof polygon in the building. It returns an array of point ids of a roof, and the size of the array in argument .I npts. It returns 0 on failure.

char *sef_get_building_name(sef_building *bld)
gets the name of the building object. It returns 0 on failure.

int sef_get_building_type(sef_building *bld)
gets the type of the building object. It returns -1 on failure.

int sef_get_building_num_attributes(sef_building *bld)
gets the number of attribute-value pairs in the building block. It returns -1 on failure.

char *sef_get_building_attribute(sef_building *bld, int i)

char *sef_get_building_value(sef_building *bld, int i)
get the ith attribute and value string from building block's attribute table. It returns 0 on failure.

Getting Information from a Surface Block

int sef_get_surface_name(sef_surface *s, char *name)
int sef_get_surface_material_type(sef_surface *s, char *name)
int sef_get_surface_function_type(sef_surface *s, char *name)
int sef_get_surface_point(sef_surface *s, sef_pt *pt)
int sef_get_surface_attributes(sef_surface *s, char *att, char *val)

Getting Information from a Constraint Block

int sef_get_constraint_name(sef_constraint *c, char *name)
int sef_get_constraint_type(sef_constraint *c, char *type)
int sef_get_constraint_attributes_linear(sef_constraint *c, double *A, double *B, double *C, double *X0, double *Y0, double *Z0)
int sef_get_constraint_attributes_planar(sef_constraint *c, double *A, double *B, double *C, double *D)
int sef_get_constraint_attributes_angle(sef_constraint *c, double *angle)

```
int sef_get_constraint_attributes(sef_constraint *c, char *att, char *val)
```

Getting Information from a Road Block

```
int sef_get_road_name(sef_road *r, char *name)
int sef_get_road_pt(sef_road *r, sef_road_pt *pt)
int sef_get_road_attributes(sef_road *r, char *att, char *val)
```

Getting Information from a Road Point Block

```
int sef_get_road_pt_name(sef_road_pt *pt, char *name)
int sef_get_road_pt_point(sef_road_pt *pt, sef_pt *sefpt)
int sef_get_road_pt_width(sef_road_pt *pt, double *width)
```

8.3 Sample program to create a site exchange file for a rectangular flatroof object

```
/*
 * An example using sitexg-api
 ***********************************************************************
 * HISTORY
 * 09-Dec-95    Yuan Hsieh (ych) at Digital Mapping Lab
 *              School of Computer Science, Carnegie Mellon University
 *              created.
 */
#include <stdio.h>
#include <sys/file.h>
#include <sitexg-api.h>
#include <time.h>

/* a list of points for a simple unit cube. */
static double points[8][3] = {
  {0.0, 0.0, 0.0},
  {1.0, 0.0, 0.0},
  {1.0, 1.0, 0.0},
  {0.0, 1.0, 0.0},
  {0.0, 0.0, 1.0},
  {1.0, 0.0, 1.0},
  {1.0, 1.0, 1.0},
  {0.0, 1.0, 1.0}
};

main( argc, argv )
int argc;
char **argv;
{
  int i;
  time_t t;
  struct tm *datev;
  sef_site *site;
  sef_building *bld;
  sef_pt *pt;
  FILE *f;

  site = sef_create();

  /* adds the file attribute information */
  sef_add_producer( site, "sef_example1" );
  sef_add_title( site, "a unit cube" );
  time( &t );
  datev = localtime( &t );
  sef_add_date( site, datev->tm_mon+1, datev->tm_mday, datev->tm_year );

  /* adds world information */
  sef_add_world_ellipsoid( site, "WGS_1984");
  sef_add_world_horizontal_datum( site, "WGS_1984");
  sef_add_world_vertical_datum( site, "MSL");

  /* adds the origin of the local coordinate system,
    * say its in the middle of Atlantic Ocean
    */
   sef_add_world_local_coord_lat( site, 'N', 42, 0, 0, 0 );
   sef_add_world_local_coord_lon( site, 'W', 40, 0, 0, 0 );
   sef_add_world_local_coord_elev( site, 0.0 );

   /* adds a list of images used to measure this cube.
    * Example:
    *
    * sef_add_world_image( site, "image_0", "tec-header_0" );
    * sef_add_world_image( site, "image_1", "tec-header_1" );
    */

   /* Adds the buildings */
   bld = sef_flatroof_create( "cube" );

   /* output the list of points */
   for ( i = 0; i < 8; i++ ) {
     pt = sef_pt_create();
     sef_add_pt_id( pt, i );
     sef_add_pt_xyz( pt, points[i][0], points[i][1], points[i][2] );

     /* the point is pretty accurate */
     sef_add_pt_covariance( pt, 1e-6, 1e-6, 1e-6, 0.0, 0.0, 0.0 );

     /* Add the image measurements for this point here
      * Example:
      *
      * int imgid;
      * double r, c, sigma;
      *
      * imgid = sef_get_world_image_id( site, "image_0" );
      * sef_add_pt_imagept( pt, imgid, r, c, sigma );
      *
      *
      */

     sef_add_building_point( bld, pt );
   }
   sef_add_building_num_floorpts( bld, 4 );
   sef_add_building_floor_elevation( bld, 0.0 );
   sef_add_building_model_height( bld, 1.0 );
   sef_add_building( site, bld );

   /* output the exchange file */
   f = fopen( "cube.exchange", "w" );
   if ( !f ) {
     printf("%s: Cannot create %s\n", argv[0], "cube.exchange" );
     exit(1);
   }
   sef_write( f, site );
   fclose( f );
   sef_free( site );
}
```

8.4 Sample program to read in and process a site exchange file

```
/*
 * An example using sitexg-api.
 **********************************************************************
 * HISTORY
 * 01-Nov-97    George Edward Bulwinkle (geb) at Digital Mapping Laboratory
 *              School of Computer Science, Carnegie Mellon University
 *              Adapted for surfaces, roads and constraints.
 *
 * 15-Nov-95    Yuan Hsieh (ych) at Digital Mapping Lab
 *              School of Computer Science, Carnegie Mellon University
 *              created.
 */
#include <stdio.h>
#include <sys/file.h>
#include <sitexg-api.h>

#define USAGE "Usage: %s <input> <output>\n"

main( argc, argv )
int argc;
char **argv;
{
  FILE *f;
  int i, nobj, npts, j;
  sef_site *site;
  sef_building *bld;
  sef_surface *surf;
  sef_road *road;
  sef_constraint *cons;
  sef_pt *pt;
  double x, y, z;

  if ( argc != 3 ) {
    printf( USAGE, argv[0] );
    exit(1);
  }

  f = fopen( argv[1], "r" );
  if ( !f ) {
    printf("%s: Cannot read %s\n", argv[0], argv[1] );
    exit(1);
  }

  site = sef_read( f );
  fclose( f );
  if ( !site ) {
    printf("%s: Error parsing %s\n", argv[0], argv[1] );
    exit(1);
  }

  nobj = site->world.nobjs;
```

```
printf("Number of objects:%d\n\n", nobj );

nobj = site->nbuilds;
printf("Number of buildings:%d\n\n", nobj );
for ( i = 0; i < nobj; i++ ) {
  bld = site->buildings[i];
  npts = sef_get_building_num_points( bld );
  printf("Building %d:", i );
  switch( bld->type ) {
  case SEF_FLATROOF:
    printf("flat roof building with %d points\n", npts);
    break;
  case SEF_RECT:
    printf("rectangular flat roof building with %d points\n", npts);
    break;
  case SEF_PEAKROOF:
    printf("peaked roof building with %d points\n", npts);
    break;
  case SEF_GENERIC:
    printf("generic roof building with %d points\n", npts);
    break;
  case SEF_OVERHANG_GENERIC:
    printf("overhang generic roof building with %d points\n", npts);
    break;
  }
  for ( j = 0; j < npts; j++ ) {
    pt = sef_get_building_point( bld, j );
    sef_get_pt_xyz( pt, &x, &y, &z );
    x += 10.0;
    y += 10.0;
    z += 10.0;
    sef_add_pt_xyz( pt, x, y, z );
  }

}

nobj = site->nsurfs;
printf("\nNumber of surfaces:%d\n\n", nobj );
for (i = 0; i < nobj; i++) {
  surf = site->surfaces[i];
  npts = sef_get_surface_num_points( surf );
  printf("Surface %d (%d points):", i, npts );
  printf("Material %s, Function %s\n", surf->mat, surf->func);

  for ( j = 0; j < npts; j++ ) {
    pt = sef_get_surface_point( surf, j );
    sef_get_pt_xyz( pt, &x, &y, &z );
    x += 10.0;
    y += 10.0;
    z += 10.0;
    sef_add_pt_xyz( pt, x, y, z );
  }

}
```

```
  nobj = site->nroads;
  printf("\nNumber of roads:%d\n\n", nobj );
  for (i = 0; i < nobj; i++) {
    road = site->roads[i];
    npts = sef_get_road_num_points( surf );
    printf("Road %d: Centerline has %d points.\n", i, npts );

    for ( j = 0; j < npts; j++ ) {
      pt = sef_get_road_pt_point( road, j );
      sef_get_pt_xyz( pt, &x, &y, &z );
      x += 10.0;
      y += 10.0;
      z += 10.0;
      sef_add_pt_xyz( pt, x, y, z );
    }

  }

  nobj = site->ncons;
  printf("\nNumber of constraints:%d\n\n", nobj );
  for (i = 0; i < nobj; i++) {
    cons = site->constraints[i];
    npts = sef_get_constraint_num_points( cons);
    printf("Constraint %d: %s containing %d points.\n",i, cons->inter, npts);

    /* Constraints dont need to be shifted */
  }




  sef_add_producer( site, "sef_example2" );
  sef_add_title( site, "shifted objects by [10,10,10]" );

  f = fopen( argv[2], "w" );
  if ( !f ) {
    printf("%s: Cannot create %s\n", argv[0], argv[2] );
    exit(1);
  }
  sef_write( f, site );
  fclose( f );
  sef_free( site );
}
```

8.5 Sample program to create a site exchange file for a surface object

```
/*
 * An example using sitexg-api to create a sefile for a surface.
 ************************************************************************
 * HISTORY
 * 01-Nov-97    George Edward Bulwinkle (geb) at Digital Mapping Lab
 *              School of Computer Science, Carnegie Mellon University
 *              created.
 */
#include <stdio.h>
#include <sys/file.h>
#include <sitexg-api.h>
#include <time.h>

/* a list of points for a simple sloping surface */
static double points[4][3] = {
  {0.5, 0.0, 0.0},
  {1.0, 0.5, 0.1},
  {0.5, 1.0, 0.1},
  {0.0, 0.5, 0.0}
};

main( argc, argv )
int argc;
char **argv;
{
  int i;
  time_t t;
  struct tm *datev;
  sef_site *site;
  sef_surface *surf;
  sef_pt *pt;
  FILE *f;

  site = sef_create();

  /* adds the file attribute information */
  sef_add_producer( site, "sef_example3" );
  sef_add_title( site, "a simple surface" );
  time( &t );
  datev = localtime( &t );
  sef_add_date( site, datev->tm_mon+1, datev->tm_mday, datev->tm_year );

  /* adds world information */
  sef_add_world_ellipsoid( site, "WGS_1984");
  sef_add_world_horizontal_datum( site, "WGS_1984");
  sef_add_world_vertical_datum( site, "MSL");

  /* adds the origin of the local coordinate system,
   * say its in the middle of Atlantic Ocean
   */
  sef_add_world_local_coord_lat( site, 'N', 42, 0, 0, 0 );
  sef_add_world_local_coord_lon( site, 'W', 40, 0, 0, 0 );
  sef_add_world_local_coord_elev( site, 0.0 );

  /* adds a list of images used to measure this cube.
   * Example:
   *
   * sef_add_world_image( site, "image_0", "tec-header_0" );
   * sef_add_world_image( site, "image_1", "tec-header_1" );
   */

  /* Adds the buildings */
  surf = sef_surface_create( "sloped-surface" );

  /* output the list of points */
  for ( i = 0; i < 4; i++ ) {
    pt = sef_pt_create();
    sef_add_pt_id( pt, i );
    sef_add_pt_xyz( pt, points[i][0], points[i][1], points[i][2] );

    /* the point is pretty accurate */
    sef_add_pt_covariance( pt, 1e-6, 1e-6, 1e-6, 0.0, 0.0, 0.0 );

    /* Add the image measurements for this point here
     * Example:
     *
     * int imgid;
     * double r, c, sigma;
     *
     * imgid = sef_get_world_image_id( site, "image_0" );
     * sef_add_pt_imagept( pt, imgid, r, c, sigma );
     *
     *
     */

    sef_add_surface_point( surf, pt );
  }
  sef_add_surface_material(surf, "concrete");
  sef_add_surface_function(surf, "walkway");
  sef_add_surface( site, surf );

  /* output the exchange file */
  f = fopen( "surface.exchange", "w" );
  if ( !f ) {
    printf("%s: Cannot create %s\n", argv[0], "cube.exchange" );
    exit(1);
  }
  sef_write( f, site );
  fclose( f );
  sef_free( site );
}
```

8.6 Hints for developing a writer/reader with the API

This section briefly presents the steps involved in developing programs that convert to and from the Site Exchange Format, using the API. Steps for developing writers (convert from X to Site Exchange) and readers (convert from Site Exchange to X) are discussed. This section is primarily intended for implementors who are familiar with basic concepts in 3D geometry, coordinate systems, and site representation, and wish to quickly develop programs using the API. It does not give complete code examples, instead stressing the API functions which will be needed to implement writers and readers.

API functions prefaced with `sef_add` are intended for use in Site Exchange writers; functions prefaced with `sef_get` are intended for use in Site Exchange readers.

8.6.1 Developing programs for conversion to Site Exchange

There are two broad steps in converting a format to Site Exchange: convert the geometric representation and store it in a placeholder object, and then call a write function on the placeholder object to write out the ASCII Site Exchange file.

```
site = sef_create();
 .
 .
/*** put stuff in site ***/
 .
 .
sef_write(fp, site);
```

Several elements of the placeholder site object must be filled in for a valid Site Exchange file to be written. First, you must set the creation information for the Site Exchange file:

```
sef_add_producer(site, "My Site Exchange Converter");
sef_add_title(site, "Title for this file");
sef_add_date(site, month, day, year);
```

Next, you must set the datum information for the coordinates of the objects you intend to store in the site file. The API provides a set of coordinate conversion functions to convert between UTM, geodetic, geocentric, and local vertical coordinates; however, all of these conversion functions assume the use of the WGS1984 datum, so if your data format is using another datum, you must carry out a datum transformation first before using the coordinate conversion functions.

Here's an example of setting the datum information:

```
sef_add_world_ellipsoid(site, "WGS_1984");
sef_add_world_horizontal_datum(site, "WGS_1984");
sef_add_world_vertical_datum(site, "MSL");
```

Next, you need to set the position of the origin of the local vertical coordinate system used by the Site Exchange File. There are two ways to do this. Either use the following three functions to set a specific geodetic location as an origin...

```
sef_add_world_local_coord_lat(site, latsgn, latdeg, latmin, latsec, lattho);
```

```
sef_add_world_local_coord_lon(site, lonsgn, londeg, lonmin, lonsec, lontho);
sef_add_world_local_coord_elevation(site, elev);
```

...or use the convenience functions **`sef_add_world_local_coord_utm()`** or **`sef_add_world_local_coord_geodetic()`**. If you don't have a particular origin in mind, it's often easiest to use a 3D point from your site model as an origin. The description of the textttreadcoords function below shows how to do this quickly.

At this point, you can begin converting your site model representation to the Site Exchange representation. Here's an example of some incomplete code for creating a flat roof building:

```
/*** Create the building placeholder. ***/
bld = sef_flatroof_create( cubename );

/*** Read in coordinates of bld from file ***/
for (i = 0; i < 8; i++) {

  /*** NOT API FUNCTIONS, JUST PSEUDO-CODE FOR READING DATA ***/
  readline(fpin);
  readcoords(site, utm_zone, &(x[i]), &(y[i]), &(z[i]));
}

/*** Store coordinates as points in the flatroof model.
     Point ordering is the same, so this is easy. ***/
for (i = 0; i < 8; i++) {
  pt = sef_pt_create();
  sef_add_pt_id(pt, i);
  sef_add_pt_xyz(pt, x[i], y[i], z[i]);

  /* the point is pretty accurate */
  sef_add_pt_covariance( pt, 1e-6, 1e-6, 1e-6, 0.0, 0.0, 0.0 );

  /* add the point to the building object */
  sef_add_building_point(bld, pt);
}

/*** Tell building how many points are floor points. ***/
sef_add_building_num_floorpts(bld, 4);

/*** Add it to the site exchange file. ***/
sef_add_building(site, bld);
```

For the sake of an example, let's assume that the input data to our converter is in UTM coordinates. Let's also assume that we want to use one of the data points in the input data as the origin for the Site Exchange file's coordinate system. The `readcoords` function might then look like this:

```
static int local_system_defined = 0;

readcoords(site, utm_zone, x, y, z)
sef_site *site;
int utm_zone;
double *x, *y, *z;
{
  double lat, lon, gx, gy, gz;

  /*** Read utm coordinates from file into x, y, z. ***/
  .....
  .....

  /*** If we haven't defined a local system, do it with the first
       point we encounter. ***/
  if (!local_system_defined) {
    sef_add_world_local_coord_utm(site, *x, *y, utm_zone, *z);
    sef_init_local_conversion();
    local_system_defined = 1;
  }

  /*** Convert from utm to local coordinate system. ***/
  sef_utm_to_geodetic(*x, *y, utm_zone, &lat, &lon);
  sef_geodetic_to_geocentric(lat, lon, *z, &gx, &gy, &gz);
  sef_geocentric_to_local(site, gx, gy, gz, x, y, z);
}
```

The idea is that `readcoords()` returns points in local coordinates, suitable for insertion into building objects. The first point encountered by `readcoords()` is used as the origin for the Site Exchange file.

8.6.2 Developing programs for conversion from Site Exchange

Developing a reader for the Site Exchange Format is just a reversal of the two major steps in developing a writer: call a read function to read the Site Exchange file into a placeholder object, and then parse out fields of the object into the desired format.

```
site = sef_read(f);
.
.
.
/*** access fields in site object through API routines ***/
.
.
/*** write output format file ***/
```

The example program in Section 8.4 illustrates the use of the API functions to obtain various components of the site model information.

9 Examples of CMU Site Exchange Format

The following section contains four examples of exchange files for actual buildings measured from the RADIUS modelboard, Fort Hood and ISPRS Avenches. Two of the files represent areas in Fort Hood. The first 3 examples demonstrate the expressive power of the exchange format for three types of buildings: an *L-shaped Flat Roof building*, a *Peak Roof building*, and an *Overhang Generic Roof building*. The final example contains a complex *L-shaped building* constructed from two *Flat Roof buildings* with *Constraints*, as well as an adjacent *Surface*.

In the case of the RADIUS modelboard building a fictional geolocation for the model was selected based upon the CMU triangulation of the modelboard imagery. This is reflected in each of the USATEC Header Interchange files associated with this Site Exchange File.

In the case of the Avenches data, the use of the **BESSEL_1841** ellipsoid is defined by the triangulation data supplied with the imagery by the ISPRS Working Group III/3
[manos@p.igp_ethz.ch].

9.1 An L–shaped Flat Roof Building

The building object in this example is shown in Figure 11. The building was measured on 8 images.

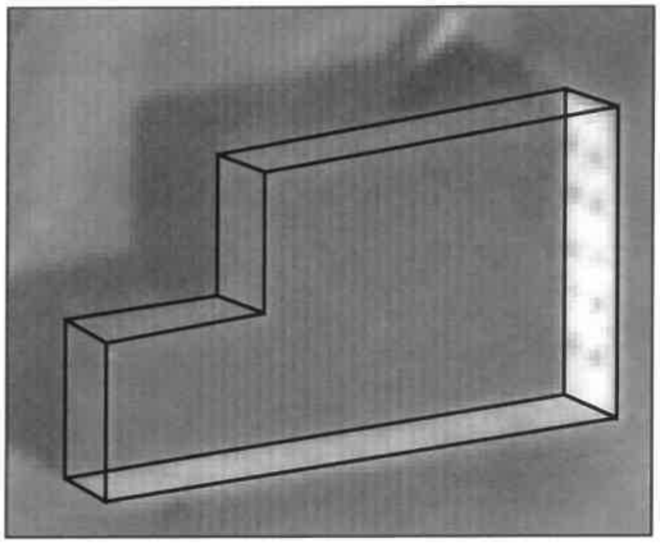

Figure 11: L–shaped building projected to image J1

```
Begin file::
  Begin file attributes::
    Producer: SiteCity 1.0
    Date: 11:13:98
    Version: CMU-Site-Exchange 5.0
    Title: flat.ste
  End file attributes
  Begin world::
    Ellipsoid Name: WGS_1984
    Horizontal Datum: WGS_1984
    Vertical Datum: MSL
    Local Origin: N 42 0 0 0 W 40 0 0 0 0.000000000000
    Geocentric to Local Matrix: 0.642787609687 0.766044443119 0.0 -0.512583782722 0.430108863030 0.74314482547
0.569281963990 -0.477684286020 0.669130606359
    Begin images::
      Number of Images: 8
      Image 0: j8
      Header 0: j8.tec
      Image 1: j7
      Header 1: j7.tec
      Image 2: j6
      Header 2: j6.tec
      Image 3: j5
      Header 3: j5.tec
      Image 4: j4
      Header 4: j4.tec
      Image 5: j3
      Header 5: j3.tec
      Image 6: j2
      Header 6: j2.tec
      Image 7: j1
      Header 7: j1.tec
    End images
    Begin attributes::
      Number of Attributes: 0
    End attributes
    Number of Objects: 1
  End world
  Begin building model::
    Model Name: E1405c6800
    Begin flat roof parameters::
      Number of Floor Points: 6
      Floor Elevation: 0.171961
      Model Height: 9.560117
    End flat roof parameters
    Begin point list::
      Number of Points: 12
      Begin point::
        Point Id: 0
        Local Coordinate: 216.195067949695 -168.041561845596 0.171936059833
        Local Covariance: 0.100779322404 0.107527200973 0.242589193243 0.020745801302 0.024188799698
0.022968210658
        Number of Image Measurements: 6
        image 1: 831.980000000000 1016.980000000000 1.000000000000
        image 2: 887.710000000000 989.970000000000 1.000000000000
        image 4: 879.460000000000 911.100000000000 1.000000000000
        image 5: 818.570000000000 216.360000000000 1.000000000000
        image 6: 151.930000000000 380.170000000000 1.000000000000
        image 7: 804.480000000000 308.680000000000 1.000000000000
      End point
      Begin point::
        Point Id: 1
        Local Coordinate: 215.812793186616 -115.728813070141 0.171995025611
        Local Covariance: 0.098239982923 0.107890085611 0.242587007570 -0.007673007805 0.028579695780
0.025074943166
        Number of Image Measurements: 7
        image 0: 729.560000000000 924.290000000000 1.000000000000
        image 1: 733.570000000000 1013.570000000000 1.000000000000
        image 2: 793.560000000000 983.030000000000 1.000000000000
        image 4: 777.580000000000 910.050000000000 1.000000000000
        image 5: 826.320000000000 313.290000000000 1.000000000000
        image 6: 248.410000000000 388.330000000000 1.000000000000
        image 7: 790.040000000000 393.560000000000 1.000000000000
      End point
      Begin point::
        Point Id: 2
        Local Coordinate: 186.876458396388 -115.940267774876 0.172017143290
        Local Covariance: 0.085408539229 0.114890095126 0.242584290969 0.021901312003 0.029699624033
0.015886471816
        Number of Image Measurements: 7
        image 0: 733.320000000000 873.120000000000 1.000000000000
        image 2: 794.940000000000 928.870000000000 1.000000000000
        image 3: 879.490000000000 932.360000000000 1.000000000000
        image 4: 777.970000000000 859.020000000000 1.000000000000
        image 5: 771.110000000000 318.610000000000 1.000000000000
        image 6: 253.770000000000 433.570000000000 1.000000000000
        image 7: 737.140000000000 393.530000000000 1.000000000000
      End point
      Begin point::
        Point Id: 3
        Local Coordinate: 187.140891996366 -152.125109871998 0.171924277558
        Local Covariance: 0.087685207575 0.116936946838 0.242594277731 0.002334506905 0.019411447269
0.014474600188
        Number of Image Measurements: 3
        image 0: 779.900000000000 868.960000000000 1.000000000000
        image 1: 803.110000000000 961.680000000000 1.000000000000
        image 3: 948.480000000000 930.280000000000 1.000000000000
      End point
      Begin point::
        Point Id: 4
        Local Coordinate: 201.053996770929 -152.023436220303 0.171954523085
        Local Covariance: 0.091780286311 0.118467665814 0.242612824788 0.004359014612 0.018706890141
-0.003690030005
        Number of Image Measurements: 1
        image 3: 949.460000000000 951.910000000000 1.000000000000
      End point
      Begin point::
        Point Id: 5
        Local Coordinate: 201.171855110048 -168.151346180847 0.171941875281
        Local Covariance: 0.089910635969 0.104627923122 0.242594425851 0.005348285030 0.024601975435
-0.004387753932
        Number of Image Measurements: 3
        image 0: 799.730000000000 889.870000000000 1.000000000000
        image 1: 833.600000000000 989.640000000000 1.000000000000
        image 3: 978.740000000000 951.200000000000 1.000000000000
      End point
      Begin point::
        Point Id: 6
        Local Coordinate: 216.195009309325 -168.041554950050 9.732129971011
        Local Covariance: 0.100773054940 0.107521125396 0.190863353732 0.020745420215 0.024975063828
0.018103253966
        Number of Image Measurements: 8
        image 0: 809.240000000000 920.640000000000 1.000000000000
        image 1: 826.990000000000 1016.310000000000 1.000000000000
        image 2: 892.190000000000 987.710000000000 1.000000000000
        image 3: 980.320000000000 984.730000000000 1.000000000000
        image 4: 880.190000000000 901.970000000000 1.000000000000
        image 5: 817.930000000000 213.940000000000 1.000000000000
        image 6: 146.130000000000 392.460000000000 1.000000000000
        image 7: 800.500000000000 299.500000000000 1.000000000000
      End point
      Begin point::
        Point Id: 7
        Local Coordinate: 215.812818339695 -115.728865901811 9.732107943431
        Local Covariance: 0.098234744595 0.107887542535 0.190864035003 -0.007672358424 0.023850106240
0.017897654830
        Number of Image Measurements: 8
        image 0: 739.630000000000 928.650000000000 1.000000000000
        image 1: 729.350000000000 1013.890000000000 1.000000000000
        image 2: 799.310000000000 980.740000000000 1.000000000000
        image 3: 882.390000000000 988.050000000000 1.000000000000
        image 4: 778.070000000000 900.910000000000 1.000000000000
        image 5: 825.700000000000 312.130000000000 1.000000000000
        image 6: 243.290000000000 399.180000000000 1.000000000000
        image 7: 786.000000000000 384.500000000000 1.000000000000
      End point
      Begin point::
        Point Id: 8
        Local Coordinate: 186.876525340297 -115.940276053900 9.732093355351
        Local Covariance: 0.085402250183 0.114887395459 0.190863850892 0.021902298229 0.023744691848
0.019217212158
        Number of Image Measurements: 8
        image 0: 743.420000000000 877.390000000000 1.000000000000
        image 1: 731.850000000000 957.710000000000 1.000000000000
        image 2: 800.190000000000 926.440000000000 1.000000000000
        image 3: 879.820000000000 941.690000000000 1.000000000000
        image 4: 778.460000000000 850.300000000000 1.000000000000
        image 5: 770.350000000000 317.470000000000 1.000000000000
        image 6: 248.690000000000 443.450000000000 1.000000000000
        image 7: 733.000000000000 384.500000000000 1.000000000000
      End point
      Begin point::
        Point Id: 9
        Local Coordinate: 187.140902852652 -152.125055936493 9.731995849518
        Local Covariance: 0.087669095544 0.116918449532 0.190863712818 0.002336523160 0.027248197235
0.019337373120
        Number of Image Measurements: 8
        image 0: 790.500000000000 872.200000000000 1.000000000000
        image 1: 799.050000000000 961.370000000000 1.000000000000
        image 2: 865.120000000000 932.220000000000 1.000000000000
        image 3: 948.960000000000 939.620000000000 1.000000000000
        image 4: 848.410000000000 852.320000000000 1.000000000000
        image 5: 766.490000000000 249.320000000000 1.000000000000
        image 6: 182.760000000000 436.780000000000 1.000000000000
        image 7: 744.500000000000 326.500000000000 1.000000000000
      End point
      Begin point::
        Point Id: 10
        Local Coordinate: 201.053970005804 -152.023404785554 9.732076618612
        Local Covariance: 0.091738513394 0.118448210001 0.190864199829 0.004361750698 0.027324609996
0.022047075606
        Number of Image Measurements: 8
        image 0: 789.710000000000 896.430000000000 1.000000000000
        image 1: 798.030000000000 987.010000000000 1.000000000000
        image 2: 865.410000000000 957.830000000000 1.000000000000
        image 3: 949.940000000000 961.280000000000 1.000000000000
        image 4: 848.970000000000 876.100000000000 1.000000000000
        image 5: 792.200000000000 246.450000000000 1.000000000000
        image 6: 179.050000000000 416.990000000000 1.000000000000
        image 7: 768.500000000000 326.500000000000 1.000000000000
      End point
      Begin point::
        Point Id: 11
        Local Coordinate: 201.171801642664 -168.151313982242 9.732064836414
        Local Covariance: 0.089872320111 0.104620959405 0.190863699977 0.005350274919 0.024942476489
0.022129791475
        Number of Image Measurements: 8
        image 0: 811.330000000000 893.640000000000 1.000000000000
        image 1: 828.610000000000 989.400000000000 1.000000000000
        image 2: 891.940000000000 959.930000000000 1.000000000000
        image 3: 979.290000000000 960.570000000000 1.000000000000
        image 4: 880.010000000000 875.780000000000 1.000000000000
        image 5: 789.540000000000 216.130000000000 1.000000000000
        image 6: 150.370000000000 413.800000000000 1.000000000000
        image 7: 773.000000000000 299.500000000000 1.000000000000
      End point
    End point list
    Begin attributes::
      Number of Attributes: 0
    End attributes
  End building model
End file
```

9.2 A Peak Roof building

The building object in this example is shown in Figure 12. The building was measured on three images.

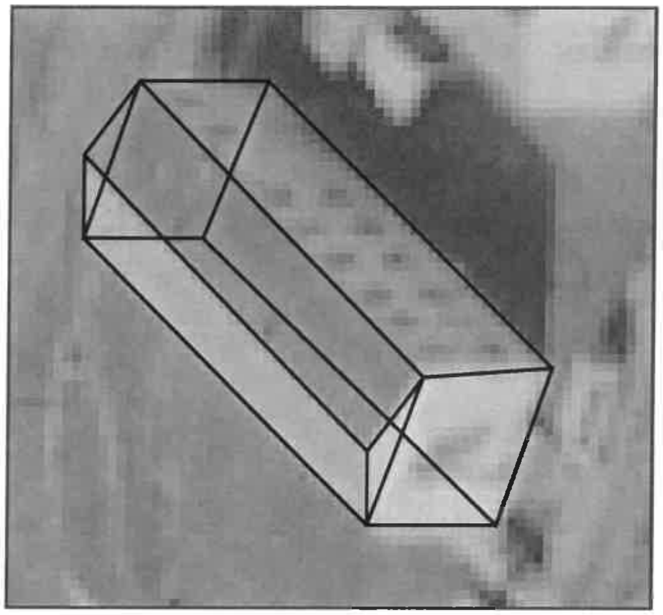

Figure 12: Peak roof building projected to image fhrad4

```
Begin file::
  Begin file attributes::
    Producer: SiteCity 1.0
    Date: 11:13:98
    Version: CMU-Site-Exchange 5.0
    Title: peak.ste
  End file attributes
  Begin world::
    Ellipsoid Name: WGS_1984
    Horizontal Datum: WGS_1984
    Vertical Datum: MSL
    Local Origin: N 31 8 33 170 W 97 45 48 216 0.000000000000
    Geocentric to Local Matrix: 0.990834347863 -0.135082549184 0.0 0.069860512419 0.512428849481 0.8558832765
-0.115614894796 -0.848038548136 0.517169041007
    Begin images::
      Number of Images: 4
      Image 0: fhrad4
      Header 0: fhrad4.tec
      Image 1: fhrad2
      Header 1: fhrad2.tec
      Image 2: fhrad3
      Header 2: fhrad3.tec
      Image 3: fhrad1
      Header 3: fhrad1.tec
    End images
    Begin attributes::
      Number of Attributes: 0
    End attributes
    Number of Objects: 1
  End world
  Begin building model::
    Model Name: E140232300
    Begin peak roof parameters::
```

```
    Floor Elevation: 287.868300
    Model Height: 6.540944
    Peak Height: 1.789389
  End peak roof parameters
  Begin point list::
    Number of Points: 10
    Begin point::
      Point Id: 0
      Local Coordinate: -305.417382284754 -255.776932094819 287.868271998067
      Local Covariance: 0.347568551490 0.128690667636 0.590448290936 0.095136214994 0.061552928699
0.264508903295
      Number of Image Measurements: 4
      image 0: 2206.650000000000 463.900000000000 0.500000000000
      image 1: 2957.750000000000 1119.530000000000 1.000000000000
      image 2: 334.040000000000 1426.020000000000 0.500000000000
      image 3: 315.870000000000 2362.140000000000 0.500000000000
    End point
    Begin point::
      Point Id: 1
      Local Coordinate: -303.955267193569 -246.697452642112 287.868343658220
      Local Covariance: 0.309397589432 0.119445145968 0.590447817183 0.065495891243 0.061067138524
0.262300956515
      Number of Image Measurements: 4
      image 0: 2189.220000000000 468.150000000000 0.500000000000
      image 1: 2931.750000000000 1117.260000000000 1.000000000000
      image 2: 348.520000000000 1441.110000000000 0.500000000000
      image 3: 328.170000000000 2378.310000000000 0.500000000000
    End point
    Begin point::
      Point Id: 2
      Local Coordinate: -330.528853399009 -242.418172116225 287.868281506693
      Local Covariance: 0.395073439159 0.123129647218 0.590441955589 0.098058220222 0.058469648661
0.239806836672
      Number of Image Measurements: 4
      image 0: 2158.850000000000 438.150000000000 0.500000000000
      image 1: 2931.930000000000 1070.170000000000 1.000000000000
      image 2: 302.360000000000 1483.170000000000 1.000000000000
      image 3: 279.790000000000 2415.750000000000 0.500000000000
    End point
    Begin point::
      Point Id: 3
      Local Coordinate: -331.990968523387 -251.497650030680 287.868302766489
      Local Covariance: 0.353534223483 0.159319193641 0.590509137316 0.072445450871 0.058949065940
0.241997960300
      Number of Image Measurements: 0
    End point
    Begin point::
      Point Id: 4
      Local Coordinate: -305.417379873576 -255.776802674954 294.409203884232
      Local Covariance: 0.347555103214 0.128662713311 0.500092310171 0.095132996146 0.062086915247
0.242314136080
      Number of Image Measurements: 4
      image 0: 2207.500000000000 450.250000000000 0.500000000000
      image 1: 2961.510000000000 1108.290000000000 1.000000000000
      image 2: 326.930000000000 1424.820000000000 0.500000000000
      image 3: 308.740000000000 2364.960000000000 0.500000000000
    End point
    Begin point::
      Point Id: 5
```

```
        Local Coordinate: -303.955285551952 -246.697291678409 294.409279206304
        Local Covariance: 0.309386254498 0.119419574820 0.500083737632 0.065497951103 0.057750845164
0.242871989455
        Number of Image Measurements: 4
        image 0: 2189.000000000000 454.500000000000 0.500000000000
        image 1: 2934.890000000000 1105.990000000000 1.000000000000
        image 2: 342.950000000000 1441.240000000000 0.500000000000
        image 3: 321.590000000000 2380.710000000000 0.500000000000
      End point
      Begin point::
        Point Id: 6
        Local Coordinate: -330.528881291301 -242.418089249550 294.409273064958
        Local Covariance: 0.395052736878 0.123105274300 0.500081761091 0.098055299890 0.057877236859
0.264985854242
        Number of Image Measurements: 4
        image 0: 2159.500000000000 425.000000000000 0.500000000000
        image 1: 2936.020000000000 1056.800000000000 1.000000000000
        image 2: 293.820000000000 1480.990000000000 0.500000000000
        image 3: 273.470000000000 2417.600000000000 0.500000000000
      End point
      Begin point::
        Point Id: 7
        Local Coordinate: -331.990975122502 -251.497596029438 294.409220891831
        Local Covariance: 0.353513211307 0.159227014752 0.500094732724 0.072451415198 0.062215220984
0.264420230096
        Number of Image Measurements: 4
        image 0: 2175.250000000000 418.000000000000 0.500000000000
        image 1: 2959.990000000000 1058.320000000000 1.000000000000
        image 2: 278.300000000000 1467.110000000000 0.500000000000
        image 3: 258.220000000000 2402.540000000000 0.500000000000
      End point
      Begin point::
        Point Id: 8
        Local Coordinate: -304.686272716090 -251.236679078815 296.198627442845
        Local Covariance: 0.322504786002 0.077304360208 0.752811531437 0.073876962741 0.059550316934
0.241924567155
        Number of Image Measurements: 4
        image 0: 2198.700000000000 449.540000000000 0.500000000000
        image 1: 2947.770000000000 1102.580000000000 1.000000000000
        image 2: 332.130000000000 1431.730000000000 1.000000000000
        image 3: 312.530000000000 2373.320000000000 0.500000000000
      End point
      Begin point::
        Point Id: 9
        Local Coordinate: -331.259867871722 -246.957472359207 296.198638169928
        Local Covariance: 0.368326109070 0.094362543308 0.752819050012 0.078817351060 0.059795944031
0.264210574956
        Number of Image Measurements: 4
        image 0: 2166.800000000000 418.160000000000 0.500000000000
        image 1: 2948.570000000000 1053.980000000000 1.000000000000
        image 2: 283.700000000000 1473.420000000000 0.500000000000
        image 3: 262.780000000000 2410.970000000000 0.500000000000
      End point
    End point list
    Begin attributes::
      Number of Attributes: 0
    End attributes
  End building model
End file
```

9.3 An Overhang Generic Roof building

The building object in this example is shown in Figure 13. The building was measured on three images.

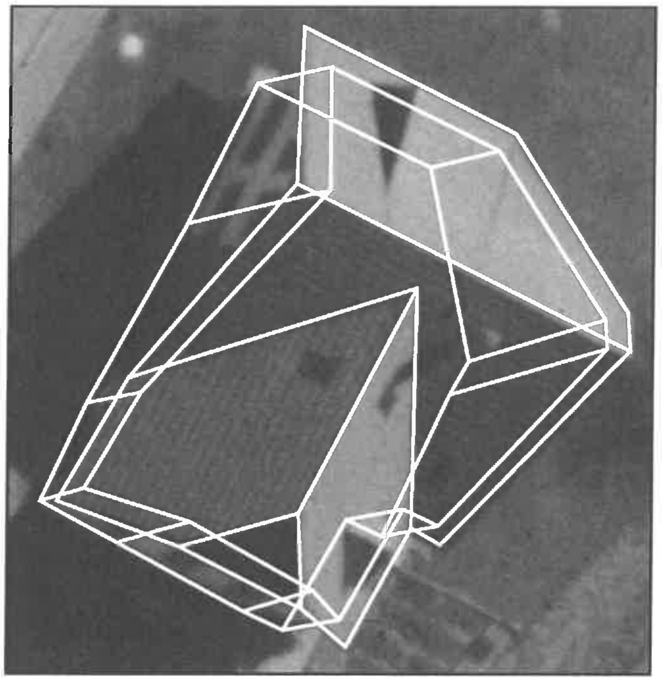

Figure 13: Overhang generic building projected to image avenches5898

```
Begin file::
  Begin file attributes::
    Producer: SiteCity 1.0
    Date: 11:13:98
    Version: CMU-Site-Exchange 5.0
    Title: Gbld.ste
  End file attributes
  Begin world::
    Ellipsoid Name: BESSEL_1841
    Horizontal Datum: BESSEL_1841
    Vertical Datum: BESSEL_1841
    Local Origin: N 46 52 49 458 E 7 2 53 887 0.000000000000
    Geocentric to Local Matrix: -0.122706044163 0.992443059689 0.0 -0.724412529632 -0.089566645648 0.683523447
0.678358101329 0.083872458301 0.729928556162
    Begin images::
      Number of Images: 4
      Image 0: avenches5898
      Header 0: avenches5898.tec
      Image 1: avenches5897
      Header 1: avenches5897.tec
      Image 2: avenches5889
      Header 2: avenches5889.tec
      Image 3: avenches5888
      Header 3: avenches5888.tec
    End images
    Begin attributes::
      Number of Attributes: 0
    End attributes
    Number of Objects: 1
  End world
  Begin building model::
    Model Name: E1403d0300
    Begin overhang generic roof parameters::
      Number of Floor Points: 12
      Number of Roof Polygons: 5
      Begin roof polygon::
        Number of Roof Points: 5
        point 0: 24
        point 1: 32
        point 2: 33
        point 3: 34
        point 4: 35
      End roof polygon
      Begin roof polygon::
        Number of Roof Points: 3
        point 0: 28
        point 1: 36
        point 2: 27
      End roof polygon
      Begin roof polygon::
        Number of Roof Points: 5
        point 0: 37
        point 1: 25
        point 2: 26
        point 3: 27
        point 4: 36
      End roof polygon
      Begin roof polygon::
        Number of Roof Points: 5
        point 0: 30
        point 1: 37
        point 2: 36
        point 3: 28
        point 4: 29
      End roof polygon
      Begin roof polygon::
        Number of Roof Points: 6
        point 0: 32
        point 1: 24
        point 2: 25
        point 3: 37
        point 4: 30
        point 5: 31
      End roof polygon
    End overhang generic roof parameters
    Begin point list::
      Number of Points: 38
      Begin point::
        Point Id: 0
        Local Coordinate: -366.366921187052 -431.697208182509 471.305578540735
        Local Covariance: 0.039585681153 0.040211858998 0.106485503198 -0.000903467621 0.003020795311
0.025232931508
        Number of Image Measurements: 2
        image 2: 1442.380000000000 1083.880000000000 1.000000000000
        image 3: 1328.880000000000 1050.530000000000 1.000000000000
      End point
      Begin point::
        Point Id: 1
        Local Coordinate: -367.700885091458 -425.862735419623 471.305647345533
        Local Covariance: 0.033871413911 0.067754496825 0.106485193167 0.004037798740 0.008312723725
0.022068931687
        Number of Image Measurements: 2
        image 2: 1517.030000000000 1045.980000000000 1.000000000000
        image 3: 1406.330000000000 1006.550000000000 1.000000000000
      End point
      Begin point::
        Point Id: 2
        Local Coordinate: -368.717041278341 -421.419807155783 471.305639139530
        Local Covariance: 0.073542930692 0.084638460673 0.106501301966 -0.004418610670 -0.000019238628
0.022516086808
        Number of Image Measurements: 3
        image 0: 1639.990000000000 987.680000000000 1.000000000000
        image 2: 1574.450000000000 1017.440000000000 1.000000000000
        image 3: 1456.870000000000 977.150000000000 1.000000000000
      End point
      Begin point::
        Point Id: 3
        Local Coordinate: -371.700406560693 -421.467025477340 471.305592717936
        Local Covariance: 1.595427610320 2.344287532540 0.106566880108 1.856914082010 0.006298777563
0.005565158018
        Number of Image Measurements: 0
      End point
      Begin point::
        Point Id: 4
        Local Coordinate: -376.266979673170 -422.486803039131 471.305593654610
        Local Covariance: 4.845952313010 0.770489754162 0.106567018928 -1.795768660020 -0.007029451100
0.018990562540
        Number of Image Measurements: 0
      End point
      Begin point::
        Point Id: 5
        Local Coordinate: -378.607460443426 -424.406286193564 471.305595585221
        Local Covariance: 0.136863066305 0.194218217095 0.106566250540 0.042054717373 0.046979108943
0.038627214451
        Number of Image Measurements: 1
        image 0: 1696.710000000000 1113.530000000000 1.000000000000
      End point
      Begin point::
        Point Id: 6
        Local Coordinate: -377.907630588243 -428.023204044327 471.305590885040
        Local Covariance: 3.153774372570 1.143723057150 0.106566871422 -1.811812737290 -0.013190503569
0.022649474080
        Number of Image Measurements: 0
      End point
      Begin point::
        Point Id: 7
        Local Coordinate: -379.319628024668 -428.333550849700 471.305591470725
        Local Covariance: 5.049195161710 3.201421795880 0.106566985004 -0.130350452275 -0.007670893510
0.027820113318
        Number of Image Measurements: 0
      End point
      Begin point::
        Point Id: 8
        Local Coordinate: -377.514569821521 -434.597999885637 471.305593964020
        Local Covariance: 3.417165129470 0.290692856207 0.106566802037 0.870055963108 0.005624824937
0.023212517908
        Number of Image Measurements: 0
      End point
      Begin point::
        Point Id: 9
        Local Coordinate: -377.048075361652 -436.228994468320 471.305592544396
        Local Covariance: 3.926409736140 4.150425464670 0.106566959128 0.471849887524 0.006080059579
0.021878100446
        Number of Image Measurements: 0
      End point
      Begin point::
        Point Id: 10
        Local Coordinate: -371.768131634681 -439.265685756516 471.305594462740
        Local Covariance: 0.153651485918 0.204632023267 0.106565471825 -0.051674593235 -0.045319400077
0.046825524451
        Number of Image Measurements: 1
        image 3: 1297.770000000000 1176.570000000000 1.000000000000
      End point
      Begin point::
        Point Id: 11
        Local Coordinate: -365.117700484968 -437.160060909412 471.305514110567
        Local Covariance: 0.090246334820 0.085447114124 0.106506470386 -0.000019058908 0.002551376438
0.027167091650
        Number of Image Measurements: 3
        image 0: 1438.210000000000 1090.520000000000 1.000000000000
        image 2: 1369.930000000000 1119.700000000000 1.000000000000
        image 3: 1255.190000000000 1088.540000000000 1.000000000000
      End point
      Begin point::
        Point Id: 12
        Local Coordinate: -366.366877878818 -431.697395674079 479.163255453764
        Local Covariance: 0.039733198290 0.040199279361 0.212016939560 -0.000856446917 -0.015835428743
0.005826473367
        Number of Image Measurements: 0
      End point
      Begin point::
        Point Id: 13
        Local Coordinate: -367.700887962168 -425.862724498714 474.501282824469
        Local Covariance: 0.033909487418 0.067859787927 0.223157528368 0.004086988247 -0.048892174866
0.002898165532
        Number of Image Measurements: 0
      End point
      Begin point::
        Point Id: 14
        Local Coordinate: -368.717038058867 -421.419805382485 474.461702368948
        Local Covariance: 0.073672439971 0.084757442635 0.224313017406 -0.004427355438 -0.008968628249
-0.037746093961
        Number of Image Measurements: 0
      End point
      Begin point::
        Point Id: 15
        Local Coordinate: -371.700405229580 -421.467019722556 476.796271559314
        Local Covariance: 1.595291637380 2.344220337870 1.739931737830 1.856968308180 -1.890670560820
-1.516960436710
        Number of Image Measurements: 0
      End point
      Begin point::
        Point Id: 16
        Local Coordinate: -376.266977838103 -422.486792290593 476.598448497605
        Local Covariance: 4.845960353150 0.770374622855 2.379848151880 -1.795855792520 -1.209158558550
3.304284332720
        Number of Image Measurements: 0
      End point
      Begin point::
        Point Id: 17
        Local Coordinate: -378.607466281739 -424.406278158244 474.436184648667
        Local Covariance: 0.136994602539 0.194361921515 0.201872973850 0.042077578673 0.071253220896
0.112252599057
        Number of Image Measurements: 0
      End point
      Begin point::
        Point Id: 18
        Local Coordinate: -377.907633982449 -428.023205661288 474.228439463674
        Local Covariance: 3.153737788060 1.143635916800 1.377164708450 -1.811894908440 -1.171004495670
1.995387743750
        Number of Image Measurements: 0
      End point
      Begin point::
        Point Id: 19
        Local Coordinate: -379.319627802861 -428.333550028590 474.199360134111
        Local Covariance: 5.049203097300 3.201375576790 2.179187934830 -0.130352391665 -2.423245057940
1.066106130280
        Number of Image Measurements: 0
      End point
      Begin point::
        Point Id: 20
        Local Coordinate: -377.514566445342 -434.598005403110 479.274272706478
        Local Covariance: 3.417175319630 0.290510910386 0.169505707088 0.870110633109 -0.008667551905
-0.005177168822
        Number of Image Measurements: 0
      End point
      Begin point::
        Point Id: 21
        Local Coordinate: -377.048071645214 -436.229001826993 477.990093557834
        Local Covariance: 3.926397958390 4.150411274700 2.363666302060 0.471871739622 2.919047799440
-0.407272273018
        Number of Image Measurements: 0
      End point
      Begin point::
        Point Id: 22
        Local Coordinate: -371.768124753592 -439.265694702589 474.717116458526
        Local Covariance: 0.153813883053 0.204736215689 0.331561994731 -0.051687474078 0.158464409089
-0.066989482565
        Number of Image Measurements: 0
      End point
      Begin point::
        Point Id: 23
        Local Coordinate: -365.117696572478 -437.160072841839 474.924896161681
        Local Covariance: 0.090359793114 0.085594520414 0.259941853344 -0.000023852264 0.062340805746
-0.015947616786
        Number of Image Measurements: 0
      End point
      Begin point::
        Point Id: 24
        Local Coordinate: -365.661959258460 -431.514229923434 479.156247065025
        Local Covariance: 0.127886557516 0.047907689990 0.228033374351 0.016013188522 -0.003762910089
0.060451928780
```

```
      Number of Image Measurements: 4
      image 0: 1486.500000000000 1111.500000000000 1.000000000000
      image 1: 1499.420000000000 1061.060000000000 1.000000000000
      image 2: 1507.930000000000 1138.400000000000 1.000000000000
      image 3: 1390.820000000000 1042.900000000000 1.000000000000
    End point
    Begin point::
      Point Id: 25
      Local Coordinate: -367.303523732194 -425.188019579744 474.065216534331
      Local Covariance: 0.064466129081 0.074790436959 0.200617272370 0.022535398113 -0.009613783181
0.015769073730
      Number of Image Measurements: 4
      image 0: 1579.500000000000 1024.500000000000 1.000000000000
      image 1: 1589.010000000000 1011.600000000000 1.000000000000
      image 2: 1546.580000000000 1057.800000000000 1.000000000000
      image 3: 1427.260000000000 1001.590000000000 1.000000000000
    End point
    Begin point::
      Point Id: 26
      Local Coordinate: -368.385927179732 -420.683918565148 474.065842690429
      Local Covariance: 0.097162180676 0.086674187125 0.185965500667 0.001491158697 -0.008254635429
0.004239463663
      Number of Image Measurements: 4
      image 0: 1637.500000000000 995.500000000000 1.000000000000
      image 1: 1644.700000000000 981.720000000000 1.000000000000
      image 2: 1610.130000000000 1024.930000000000 1.000000000000
      image 3: 1483.470000000000 970.110000000000 1.000000000000
    End point
    Begin point::
      Point Id: 27
      Local Coordinate: -371.468875938341 -421.183548695307 476.562668927358
      Local Covariance: 0.097163302071 0.086041832509 0.188165312783 0.002294962393 -0.008250722759
0.006320311329
      Number of Image Measurements: 4
      image 0: 1657.000000000000 1051.000000000000 1.000000000000
      image 1: 1669.370000000000 1017.850000000000 1.000000000000
      image 2: 1652.890000000000 1084.580000000000 1.000000000000
      image 3: 1529.530000000000 1010.220000000000 1.000000000000
    End point
    Begin point::
      Point Id: 28
      Local Coordinate: -376.364362987101 -422.449624787385 476.532327903427
      Local Covariance: 0.147009435592 0.085596450066 0.215419917359 0.006718600468 0.016303904073
0.114985381161
      Number of Image Measurements: 4
      image 0: 1689.000000000000 1114.000000000000 1.000000000000
      image 1: 1702.540000000000 1082.260000000000 1.000000000000
      image 2: 1685.890000000000 1149.410000000000 1.000000000000
      image 3: 1560.840000000000 1071.290000000000 1.000000000000
    End point
    Begin point::
      Point Id: 29
      Local Coordinate: -379.295737667921 -423.433140207547 474.113778247722
      Local Covariance: 0.148619730808 0.086787993475 0.221633529821 0.007804223199 0.017186998075
0.116877825331
      Number of Image Measurements: 4
      image 0: 1708.500000000000 1133.500000000000 1.000000000000
      image 1: 1723.550000000000 1122.710000000000 1.000000000000
      image 2: 1680.210000000000 1170.520000000000 1.000000000000
```

```
          image 3: 1555.880000000000 1109.220000000000 1.000000000000
        End point
        Begin point::
          Point Id: 30
          Local Coordinate: -378.313596156447 -427.782259155804 473.970375487618
          Local Covariance: 0.131474163428 0.075456591270 0.164082122660 -0.005649319650 -0.023220496202
0.071965225522
          Number of Image Measurements: 4
          image 0: 1653.500000000000 1163.000000000000 1.000000000000
          image 1: 1669.040000000000 1152.060000000000 1.000000000000
          image 2: 1620.980000000000 1198.360000000000 1.000000000000
          image 3: 1500.140000000000 1140.930000000000 1.000000000000
        End point
        Begin point::
          Point Id: 31
          Local Coordinate: -379.722085828672 -428.112817675970 473.957278223197
          Local Covariance: 0.135401488212 0.080430453842 0.199625003481 -0.001433389028 -0.033894902468
0.064006342096
          Number of Image Measurements: 4
          image 0: 1662.500000000000 1180.500000000000 1.000000000000
          image 1: 1678.690000000000 1171.430000000000 1.000000000000
          image 2: 1633.600000000000 1216.500000000000 1.000000000000
          image 3: 1508.750000000000 1157.770000000000 1.000000000000
        End point
        Begin point::
          Point Id: 32
          Local Coordinate: -378.089582697570 -434.747665139682 479.279982965833
          Local Covariance: 0.127360087949 0.072828502589 0.178241319965 0.011574891967 0.004326711935
0.047546331185
          Number of Image Measurements: 4
          image 0: 1569.000000000000 1270.500000000000 1.000000000000
          image 1: 1587.660000000000 1222.530000000000 1.000000000000
          image 2: 1589.540000000000 1304.080000000000 1.000000000000
          image 3: 1471.330000000000 1205.670000000000 1.000000000000
        End point
        Begin point::
          Point Id: 33
          Local Coordinate: -377.505007212231 -436.596311137978 477.813281710419
          Local Covariance: 0.122797170448 0.084659463820 0.181331458830 0.018511079984 0.040625537201
0.038345287543
          Number of Image Measurements: 4
          image 0: 1547.500000000000 1270.000000000000 1.000000000000
          image 1: 1561.060000000000 1234.450000000000 1.000000000000
          image 2: 1553.310000000000 1301.680000000000 1.000000000000
          image 3: 1434.380000000000 1215.460000000000 1.000000000000
        End point
        Begin point::
          Point Id: 34
          Local Coordinate: -371.690073426437 -440.176850956102 474.036210429462
          Local Covariance: 0.128247647643 0.085687799075 0.251880199700 0.016214217679 0.033890203567
0.058553072088
          Number of Image Measurements: 4
          image 0: 1462.000000000000 1212.500000000000 1.000000000000
          image 1: 1475.440000000000 1203.840000000000 1.000000000000
          image 2: 1423.980000000000 1242.960000000000 1.000000000000
          image 3: 1308.640000000000 1182.620000000000 1.000000000000
        End point
        Begin point::
          Point Id: 35
        Local Coordinate: -363.977400586262 -438.185831847640 473.947948315031
        Local Covariance: 0.128870709825 0.085798964687 0.276323508035 0.014847223060 0.030531342918
0.063726664689
        Number of Image Measurements: 4
        image 0: 1409.500000000000 1111.500000000000 1.000000000000
        image 1: 1420.830000000000 1103.490000000000 1.000000000000
        image 2: 1374.160000000000 1140.270000000000 1.000000000000
        image 3: 1260.180000000000 1086.570000000000 1.000000000000
      End point
      Begin point::
        Point Id: 36
        Local Coordinate: -373.420308394618 -423.572188240214 478.531172247866
        Local Covariance: 0.108935992647 0.085967886232 0.194965077360 0.001718898801 0.004008699374
0.055749402979
        Number of Image Measurements: 4
        image 0: 1645.500000000000 1110.500000000000 1.000000000000
        image 1: 1659.370000000000 1064.050000000000 1.000000000000
        image 2: 1664.390000000000 1143.010000000000 1.000000000000
        image 3: 1540.950000000000 1051.320000000000 1.000000000000
      End point
      Begin point::
        Point Id: 37
        Local Coordinate: -371.287123056713 -431.952607088053 478.437018850027
        Local Covariance: 0.121598600728 0.077103914527 0.140135988419 -0.002579350549 -0.038241618475
0.048372510550
        Number of Image Measurements: 4
        image 0: 1536.500000000000 1166.500000000000 1.000000000000
        image 1: 1550.250000000000 1124.660000000000 1.000000000000
        image 2: 1553.250000000000 1196.260000000000 1.000000000000
        image 3: 1431.520000000000 1107.190000000000 1.000000000000
      End point
    End point list
    Begin attributes::
      Number of Attributes: 0
    End attributes
  End building model
End file
```

9.4 An L–shaped complex Building and Surface

The building object in this example is shown in Figure 14. The building was measured on 4 images. The complex building is constructed from two *Flat Roof buildings* constrained together by coplanar floors and coplanar walls. The adjacent surface is an Asphalt Parking Lot.

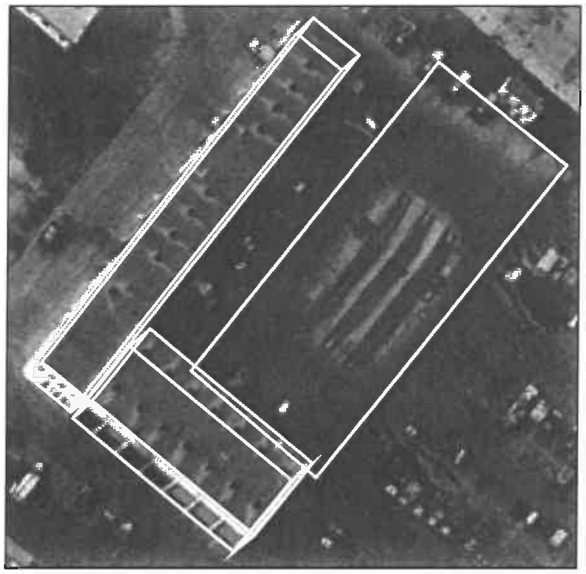

Figure 14: L–shaped building and Surface projected to image fhrad1

```
Begin file::
  Begin file attributes::
    Producer: SiteCity 1.0
    Date: 11:12:97
    Version: CMU-Site-Exchange 5.0
    Title: radt9_doc.ste
  End file attributes
  Begin world::
    Ellipsoid Name: WGS_1984
    Horizontal Datum: WGS_1984
    Vertical Datum: MSL
    Local Origin: N 31 8 33 170 W 97 45 48 216 0.000000001863
    Geocentric to Local Matrix: 0.990834347863 -0.135082549184 0.0 0.069860512419 0.512428849481 0.8558832765
-0.115614894796 -0.848038548136 0.517169041007
    Begin images::
      Number of Images: 4
      Image 0: fhrad4
      Header 0: fhrad4.tec
      Image 1: fhrad2
      Header 1: fhrad2.tec
      Image 2: fhrad3
      Header 2: fhrad3.tec
      Image 3: fhrad1
      Header 3: fhrad1.tec
    End images
    Begin attributes::
      Number of Attributes: 0
    End attributes
```

```
      Number of Objects: 5
    End world
    Begin constraint::
      name: 0x4007d060
      type: COPLANAR
      A:0 B:0 C:0 D:0
      npts: 8
      pt 0: r9-19-int 3
      pt 1: r9-19-int 0
      pt 2: r9-19-int 4
      pt 3: r9-19-int 7
      pt 4: r9-17-int 1
      pt 5: r9-17-int 2
      pt 6: r9-17-int 6
      pt 7: r9-17-int 5
      Begin attributes::
        Number of Attributes: 0
      End attributes
    End constraint
    Begin constraint::
      name: 0x4008d560
      type: COPLANAR
      A:0 B:0 C:0 D:0
      npts: 8
      pt 0: r9-19-int 0
      pt 1: r9-19-int 3
      pt 2: r9-19-int 2
      pt 3: r9-19-int 1
      pt 4: r9-17-int 0
      pt 5: r9-17-int 3
      pt 6: r9-17-int 2
      pt 7: r9-17-int 1
      Begin attributes::
        Number of Attributes: 0
      End attributes
    End constraint
    Begin building model::
      Model Name: r9-17-int
      Begin flat roof parameters::
        Number of Floor Points: 4
        Floor Elevation: 292.479649
        Model Height: 6.576665
      End flat roof parameters
      Begin point list::
        Number of Points: 8
        Begin point::
          Point Id: 0
          Local Coordinate: -425.433246897795 183.867462562406 292.479651588486
          Local Covariance: 0.668434271822 0.188113023114 0.287977633856 -0.027483405044 -0.038867207659
0.177444147163
          Number of Image Measurements: 4
          image 0: 1203.200000000000 530.140000000000 0.630000000000
          image 1: 1979.330000000000 691.620000000000 0.630000000000
          image 2: 636.350000000000 2427.360000000000 0.630000000000
          image 3: 567.830000000000 3378.980000000000 0.630000000000
        End point
        Begin point::
          Point Id: 1
          Local Coordinate: -423.335921493621 196.096123528405 292.479660901847
          Local Covariance: 0.651083507509 0.087881281052 0.287960787755 -0.075520599101 -0.041274296113
0.176962859303
          Number of Image Measurements: 4
          image 0: 1178.880000000000 538.680000000000 0.630000000000
          image 1: 1952.250000000000 689.030000000000 0.630000000000
          image 2: 654.440000000000 2447.640000000000 0.630000000000
          image 3: 584.680000000000 3399.920000000000 0.630000000000
        End point
        Begin point::
          Point Id: 2
          Local Coordinate: -507.766218714050 208.878340393405 292.479648388803
          Local Covariance: 0.752727314645 0.194826549788 0.287999122872 -0.040657406762 -0.040720620959
0.189494653450
          Number of Image Measurements: 2
          image 0: 1110.070000000000 440.090000000000 0.630000000000
          image 2: 504.240000000000 2578.610000000000 0.630000000000
        End point
        Begin point::
          Point Id: 3
          Local Coordinate: -509.647493496733 197.123244645815 292.479634886091
          Local Covariance: 0.750320662897 0.197591133893 0.288015241027 -0.035707379097 -0.043086144603
0.190187888714
          Number of Image Measurements: 2
          image 1: 2006.660000000000 564.560000000000 0.630000000000
          image 3: 410.110000000000 3502.920000000000 0.630000000000
        End point
        Begin point::
          Point Id: 4
          Local Coordinate: -425.433258875659 183.867470693204 299.056317222255
          Local Covariance: 0.668445517203 0.188114144988 0.460736781010 -0.027488411352 -0.054042782340
0.246302145256
          Number of Image Measurements: 4
          image 0: 1199.600000000000 516.490000000000 0.630000000000
          image 1: 1979.390000000000 678.220000000000 0.630000000000
          image 2: 630.190000000000 2429.490000000000 0.630000000000
          image 3: 561.630000000000 3386.080000000000 0.630000000000
        End point
        Begin point::
          Point Id: 5
          Local Coordinate: -423.335928777205 196.096163222430 299.056307235202
          Local Covariance: 0.651095463515 0.087875948989 0.460736613902 -0.075524800765 -0.049781272163
0.249088317233
          Number of Image Measurements: 4
          image 0: 1175.180000000000 525.040000000000 0.630000000000
          image 1: 1952.190000000000 675.620000000000 0.630000000000
          image 2: 648.360000000000 2449.870000000000 0.630000000000
          image 3: 578.560000000000 3407.130000000000 0.630000000000
        End point
        Begin point::
          Point Id: 6
          Local Coordinate: -507.766225280647 208.878377209305 299.056315509417
          Local Covariance: 0.752737287547 0.194821108182 0.460736470735 -0.040660844921 -0.059516931716
0.267517759875
          Number of Image Measurements: 4
          image 0: 1106.180000000000 426.410000000000 0.630000000000
          image 1: 1980.450000000000 548.570000000000 0.630000000000
          image 2: 497.390000000000 2581.510000000000 0.630000000000
          image 3: 420.110000000000 3531.750000000000 0.630000000000
        End point
        Begin point::
          Point Id: 7
          Local Coordinate: -509.647504593449 197.123255672128 299.056317516384
          Local Covariance: 0.750330581754 0.197585146568 0.460736568589 -0.035711756887 -0.059619385926
0.266975838043
          Number of Image Measurements: 4
          image 0: 1130.080000000000 418.280000000000 0.630000000000
          image 1: 2006.840000000000 551.010000000000 0.630000000000
          image 2: 479.150000000000 2561.100000000000 0.630000000000
          image 3: 403.100000000000 3510.670000000000 0.630000000000
        End point
      End point list
      Begin attributes::
        Number of Attributes: 0
      End attributes
    End building model
    Begin building model::
      Model Name: r9-19-int
      Begin flat roof parameters::
        Number of Floor Points: 4
        Floor Elevation: 292.479756
        Model Height: 7.690200
      End flat roof parameters
      Begin point list::
        Number of Points: 8
        Begin point::
          Point Id: 0
          Local Coordinate: -423.541902066315 196.127533235819 292.479733674870
          Local Covariance: 0.658604661809 0.088905776586 0.287963459203 -0.080928404831 -0.040653428480
0.172903526378
          Number of Image Measurements: 4
          image 0: 1176.840000000000 537.400000000000 1.000000000000
          image 1: 1950.780000000000 689.500000000000 1.000000000000
          image 2: 655.390000000000 2449.200000000000 0.630000000000
          image 3: 587.560000000000 3400.680000000000 2.000000000000
        End point
        Begin point::
          Point Id: 1
          Local Coordinate: -416.172628356418 238.418652615036 292.479768322733
          Local Covariance: 0.682036100424 0.207013409088 0.288023614132 -0.050506970605 -0.046308122549
0.172381015069
          Number of Image Measurements: 4
          image 0: 1088.810000000000 571.120000000000 1.000000000000
          image 1: 1852.880000000000 680.790000000000 1.000000000000
          image 2: 721.570000000000 2521.930000000000 0.630000000000
          image 3: 649.260000000000 3475.790000000000 2.000000000000
        End point
        Begin point::
          Point Id: 2
          Local Coordinate: -434.451300036153 241.683427612606 292.479809349610
          Local Covariance: 0.698976139058 0.207406936381 0.288016085395 -0.050760567622 -0.046398703692
0.177071591100
          Number of Image Measurements: 2
          image 0: 1072.970000000000 547.070000000000 0.630000000000
          image 2: 690.140000000000 2549.950000000000 0.630000000000
        End point
        Begin point::
          Point Id: 3
          Local Coordinate: -441.120840540765 198.788868515323 292.479712995099
        Local Covariance: 0.751970260733 0.070483003740 0.287957026595 -0.107618470195 -0.041641736469
0.182795795326
        Number of Image Measurements: 2
        image 1: 1956.490000000000 661.940000000000 1.000000000000
        image 3: 554.500000000000 3427.200000000000 2.000000000000
      End point
      Begin point::
        Point Id: 4
        Local Coordinate: -423.541865924142 196.127596336677 300.169954578168
        Local Covariance: 0.658615112369 0.088906343015 0.475476400967 -0.080930355072 -0.056710537031
0.252009265861
        Number of Image Measurements: 4
        image 0: 1173.180000000000 523.950000000000 1.000000000000
        image 1: 1949.710000000000 673.280000000000 0.630000000000
        image 2: 649.410000000000 2451.410000000000 0.630000000000
        image 3: 579.540000000000 3409.790000000000 0.630000000000
      End point
      Begin point::
        Point Id: 5
        Local Coordinate: -416.172606365414 238.418706562473 300.169943486485
        Local Covariance: 0.682045295095 0.207009330399 0.475477071058 -0.050512558255 -0.069145853224
0.248991122532
        Number of Image Measurements: 4
        image 0: 1084.740000000000 557.700000000000 1.000000000000
        image 1: 1851.380000000000 664.550000000000 0.630000000000
        image 2: 715.920000000000 2524.490000000000 0.630000000000
        image 3: 641.550000000000 3485.280000000000 0.630000000000
      End point
      Begin point::
        Point Id: 6
        Local Coordinate: -434.451219919156 241.683368299098 300.169923423345
        Local Covariance: 0.698985166675 0.207401046804 0.475475873694 -0.050767178860 -0.069920527572
0.255361427503
        Number of Image Measurements: 4
        image 0: 1068.860000000000 533.650000000000 0.630000000000
        image 1: 1857.750000000000 637.080000000000 0.630000000000
        image 2: 684.330000000000 2552.660000000000 0.630000000000
        image 3: 608.390000000000 3511.960000000000 0.630000000000
      End point
      Begin point::
        Point Id: 7
        Local Coordinate: -441.120883949154 198.788980739819 300.170004443029
        Local Covariance: 0.751957002791 0.070478708358 0.475480215844 -0.107609490336 -0.059030448539
0.280889083408
        Number of Image Measurements: 4
        image 0: 1158.380000000000 501.240000000000 1.000000000000
        image 1: 1955.450000000000 645.690000000000 0.630000000000
        image 2: 617.770000000000 2479.550000000000 0.630000000000
        image 3: 546.310000000000 3436.450000000000 0.630000000000
      End point
    End point list
    Begin attributes::
      Number of Attributes: 0
    End attributes
  End building model
  Begin surface model::
    name: St102956c0_879319245
    material: Asphalt
    function: Parking Lot
    Begin point list::
      Number of Points: 4
      Begin point::
        Point Id: 0
        Local Coordinate: -436.919512583755 247.015353806083 291.338786004234
        Local Covariance: 3.657995077660 0.598180887298 3.941236965970 -0.705601362922 -0.859536374920
3.176131223010
        Number of Image Measurements: 4
        image 0: 1059.530000000000 551.450000000000 1.000000000000
        image 1: 1846.670000000000 649.810000000000 1.000000000000
        image 2: 692.500000000000 2561.500000000000 1.000000000000
        image 3: 614.900000000000 3513.510000000000 1.000000000000
      End point
      Begin point::
        Point Id: 1
        Local Coordinate: -514.334593277635 259.186725893015 291.421826779001
        Local Covariance: 4.205663815170 0.623435531368 4.039314364030 -0.810786290367 -0.918745272816
3.488849032360
        Number of Image Measurements: 4
        image 0: 1002.120000000000 459.220000000000 1.000000000000
        image 1: 1876.890000000000 534.830000000000 1.000000000000
        image 2: 556.500000000000 2684.500000000000 1.000000000000
        image 3: 471.020000000000 3628.660000000000 1.000000000000
      End point
      Begin point::
        Point Id: 2
        Local Coordinate: -519.231154773116 226.881597603314 291.957452244344
        Local Covariance: 4.191926203140 0.569227739023 4.001126238610 -0.702203912366 -0.809168301782
3.470375274250
        Number of Image Measurements: 4
        image 0: 1065.580000000000 435.980000000000 1.000000000000
        image 1: 1945.590000000000 541.260000000000 1.000000000000
        image 2: 503.000000000000 2626.500000000000 1.000000000000
        image 3: 423.390000000000 3568.530000000000 1.000000000000
      End point
      Begin point::
        Point Id: 3
        Local Coordinate: -442.578911129375 215.248300415094 291.226642144084
        Local Covariance: 3.653669316750 0.548161645546 3.910072933610 -0.609324509861 -0.755646221094
3.166356836670
        Number of Image Measurements: 4
        image 0: 1122.230000000000 528.820000000000 1.000000000000
        image 1: 1919.480000000000 655.560000000000 1.000000000000
        image 2: 642.000000000000 2504.500000000000 1.000000000000
        image 3: 565.620000000000 3455.660000000000 1.000000000000
      End point
    End point list
    Begin attributes::
      Number of Attributes: 0
    End attributes
  End surface model
End file
```

10 USATEC Exchange Format

The following is an example of an USATEC interchange file for the Fort Hood image FHN711 distributed as a part of the RADIUS test dataset. It shows the complete USATEC header that defines a rigorous frame camera model. A copy of the documentation for the USATEC interchange rationale and format are available as TEC.TEX and TEC.PS in the distribution containing this document.

```
fhn711.tec
 8
7708
7688
0
255
1
     1.000       0.000       0.000
     0.000       1.000       0.000

     1.000       0.000
     0.000      -1.000
    30.000         0.000       0.000

        0.9999999776
       -0.0058666667
        0.0000000000
     -115.2005767822
        0.0058000000
        0.9999999776
        0.0000000000
      114.3705978394

0.0 0.0 0.0
0.0 0.0 0.0
0.0 0.0 0.0
0.0 0.0 0.0

0.0 30.0 40.0
50.0 60.0 70.0
80.0 90.0 100.0
110.0 120.0 140.0

0.0

  153.4740

    0.9884522584    -0.1512119586     0.0098527415
    0.1511466111     0.9884860424     0.0070743125
   -0.0108090181    -0.0055034117     0.9999264361

     596.852      -464.830      1907.867

           6368173.8

WGS_1984
```

```
   -97.   45.   48.216
    31.    8.   33.170
      0.000

0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0

  617881.739  3446057.879  14

0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0

covar-6

   0.3954845797   -0.1476719727   -0.0410932250    0.0000836969    0.0002053293   -0.0000046096
  -0.1476719727    0.4164936974   -0.0075914903   -0.0002185256   -0.0000856580    0.0000017141
  -0.0410932250   -0.0075914903    0.0408296248    0.0000043027   -0.0000217746    0.0000007536
   0.0000836969   -0.0002185256    0.0000043027    0.0000001174    0.0000000484   -0.0000000002
   0.0002053293   -0.0000856580   -0.0000217746    0.0000000484    0.0000001097   -0.0000000019
  -0.0000046096    0.0000017141    0.0000007536   -0.0000000002   -0.0000000019    0.0000000121

date 10:25:1993

gmt  16:42:44

utm-zone   14
```

11 FBIP Sensor Exchange Format

The following is an example of an FBIP file for the Fort Hood image FHN711 distributed as a part of the RADIUS test dataset. It shows the complete FBIP header and a *small* portion of the coefficients that define the piecewise approximation to the rigorous sensor model. A copy of the documentation written by Mark Horwedel of LMC including the FBIP technical rationale and format are available as FBIP.TEX and FBIP.PS in the distribution directory containing this document.

```
RADIUS BLOCK INTERPOLATION PROJECTION
IMAGE-ORIGIN  (     0.0000      0.0000)
BLOCK-SIZE  (  256.0000   256.0000)
N-BLOCKS (         31          31)
SCALE-FACTOR  3600.0000
VERTICAL-UNITS  METERS
ELEVATION-LEVELS  (     150.000      250.000      350.000  )
WORLD-COORDINATE-SYSTEM  LONG-LAT
 ORIGIN-LAT-LONG (         0.0000000000          0.0000000000)
REFERENCE-ELLIPSOID WGS-84
DATA-FORMAT ASCII
DATE 10:25:1993
GMT 16:42:44
HEADER-END
  -351980.53570992   112062.49257221  -351977.41495167   112064.51802798  -351974.29430688   112066.54336471
  -351977.31630907   112062.06493209  -351974.37877315   112064.11468616  -351971.44134047   112066.16432461
  -351974.09382515   112061.63683046  -351971.33968160   112063.71091113  -351968.58563171   112065.78487935
  -351970.86824727   112061.20826574  -351968.29766709   112063.30670146  -351965.72717159   112065.40502763
  -351967.63956452   112060.77923633  -351965.25271966   112062.90205567  -351962.86595107   112065.02476812
  -351964.40776594   112060.34974061  -351962.20482931   112062.49697230  -351960.00196105   112064.64409949
  -351961.17284052   112059.91977697  -351959.15398599   112062.09144988  -351957.13519241   112064.26302041
  -351957.93477721   112059.48934380  -351956.10017961   112061.68548694  -351954.26563599   112063.88152954
  -351954.69356491   112059.05843945  -351953.04340004   112061.27908199  -351951.39328260   112063.49962554
  -351951.44919246   112058.62706230  -351949.98363711   112060.87223355  -351948.51812298   112063.11730706
  -351948.20164868   112058.19521070  -351946.92088061   112060.46494011  -351945.64014788   112062.73457274
  -351944.95092232   112057.76288301  -351943.85512027   112060.05720017  -351942.75934797   112062.35142123
  -351941.69700208   112057.33007755  -351940.78634581   112059.64901223  -351939.87571390   112061.96785116
  -351938.43987663   112056.89679267  -351937.71454687   112059.24037477  -351936.98923628   112061.58386114
  -351935.17953459   112056.46302670  -351934.63971306   112058.83128627  -351934.09990569   112061.19944982
```

References

[1] M. S. Horwedel. A generic sensor model.

[2] Y. Hsieh. Design and evaluation of a semi-automated site modeling system. Technical Report CMU–CS–95–195, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, November 1995.

[3] J. C. McGlone and J. A. Shufelt. Projective and object space geometry for monocular building extraction. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 54–61, Seattle, Washington, 19–23 June 1994.

[4] F. R. Norvelle. Digital photogrammetric compilation package, example of an image header file, 1991.

[5] F. R. Norvelle. Digital stereoimage transformation procedures, 1991.

[6] J. Shufelt. Exploiting photogrammetric methods for building extraction in aerial images. In *International Archives of Photogrammetry and Remote Sensing*, volume XXXI, B6, S, pages 74–79, 1996.

[7] Chester C. Slama, editor. *Manual of Photogrammetry*. American Society of Photogrammetry, fourth edition, 1980.