Multiple Model Photo Popup

Jonathan Bidwell, Justin DeCell, Stephen Mokszycki
Rensselaer Polytechnic Institute, Troy, New York

This paper presents an application that enables users to align three dimensional models generated from their digital photographs and Photo Popup. This software provides both a manual and automatic method for aligning three dimensional models to re-create scenes. From photos of historic architecture to your favorite vacation destinations this software enables users to transform digital photos into three dimensional models that can be aligned to form large scenes.

Introduction

Building 3D models from collections of photographs has long since been possible using projective geometry. Estimates of unknown parameters often restrict these techniques to work under controlled settings such as with images taken from a calibrated camera. In recent years [3] [4] have introduced user-interaction and automatic image segmentation techniques to reduce these constraints.

This paper presents builds upon these techniques and presents a software application to make transforming collections of 2D photographs into 3D models mainstream. By enabling users to generate three dimensional models from two-dimensional still images this software provides manual and automatic alignment options that enable users to create large three-dimensional scenes from multiple photographs.

Based on the work of Hoiem's and Efros Automatic Photo Pop-up this software is described in terms of three subsystems: a framework, interactive molding program and algorithm for aligning three dimensional models created using an implementation of Photo Popup[4]. First a framework is presented to automate the process of creating three dimensional models from digital photos. To align these 3D models a graphical user interface (GUI) is described that enables users to manually align 3D models and projective textures. Finally an iterative closest point algorithm is applied to refine the model alignment.

Segmenting Images

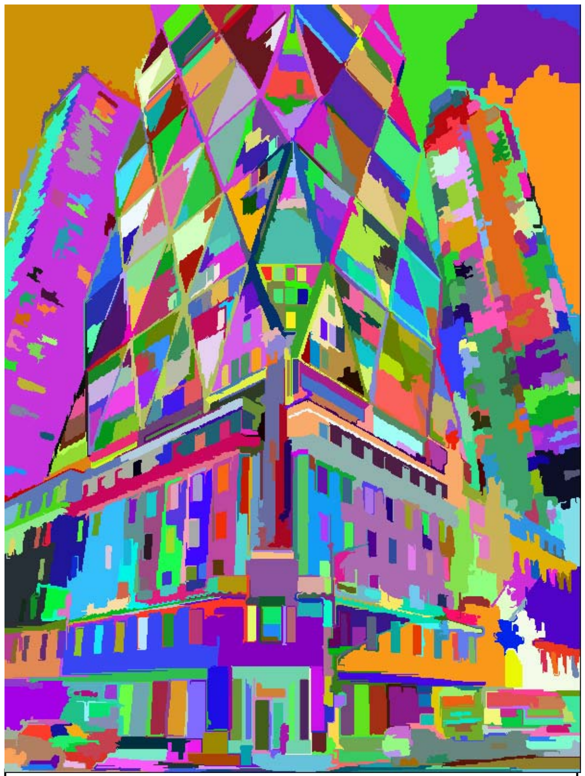


Fig S1: *This is an image of the Hearst building after being segmented into groups of* ***super pixels***

Image segmentation is a large field of research in its own right, with a wide range of ideas and uses. We will provide a basic overview of the image segmentation method used in our program.

The basic idea behind segmentation is to break an image into similar clusters of data for later use. These clusters are often called ***super pixels***, or groups of pixels with similar predefined attributes that are closely related. "In general, image segmentation techniques generally represent the problem in terms of a graph G = (V,E) where each node $v_i \in V$ corresponds to a pixel in the image, and the edges in E connect certain pairs of neighboring pixels."[Felzenszwalb] The weight of an edge is determined by the strength of the correlation of the attributes between two neighboring pixel and hence the chances the two pixels will be in the same super pixel. By keeping the strongest edges together, within a predefined limit, the image can be segmented into small or large super pixels. These super pixels will be used later to construct a 3D model. Faces of the resulting mesh will be formed based on the clumping of these super pixels.

This project uses the segmentation algorithm presented by Felzenszwalb & Huttenlocher, which breaks an image into many small groups of super pixels, and then assigns a color to each super pixel, as in Figure S1. This particular algorithm was chosen because it was relatively simple to follow and had a run time of O(n log n), where n is the number of pixels in an image. Compared to other algorithms this is really fast. They also broke out of the box decided to form and break edges based on how *dissimilar* each pixel is in relation to its neighbors. This dissimilarity is based on differences in intensity, color, motion, location, and a random local attribute. This particular algorithm has a limited number of parameters to adjust: a constant used to keep the segmentation in bounds of the image, the minimum radius of inclusion (distance pixels can be apart), and Sigma, the amount of temporarily blur applied to a pixel. A Gaussian blur applied to the image for the dissimilar comparison.

Constellations to 3D Images:

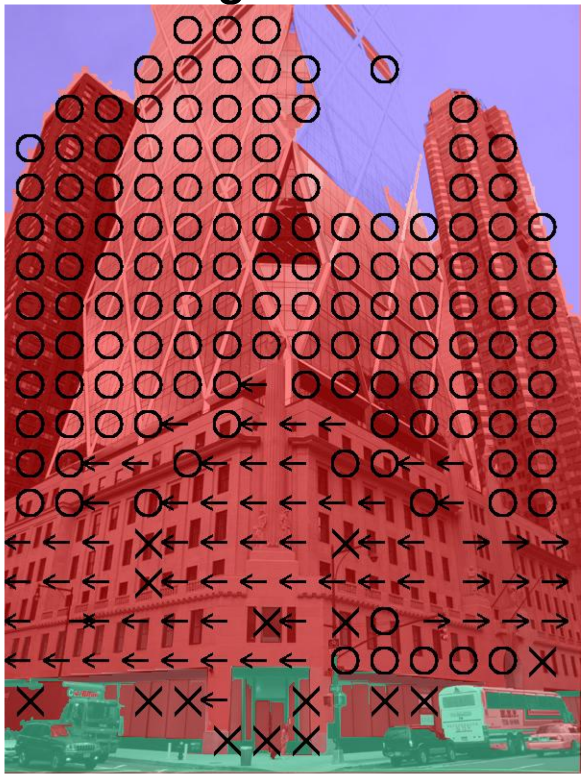


Fig S2: Constellation map of the Hearst Building. Green region = Ground, Red Vertical, Blue Sky. Arrows indicate fold directions, O what should go together, and X what did not match well.

Constellations are the next fundamental step, and are created and implemented in Photo Popup. A constellation is a super pixel of super pixels. It is a grouping of similar super pixels based on multiple attributes including color, location, shape, texture, and 3D geometry. As you can see this is starting to look like another graph theory algorithm, and that is because it is an altered version of the F&H algorithm, with far more variables taken into account. Each variable that was mentioned above has between four and eight sub sections that are taken into consideration. By matching the *similarities* between these attributes, Photo Popup attempts to group super pixels together such that they form a single object with in the picture i.e. a tree, car, building, sky, ground, etc. Photo popup then generalize the image even further by classifying each constellation into one of three categories, sky, vertical (object), or ground, using the combined averages of each variable data set.

After the computation of constellations there is finally enough data to start generating a 3D model. From the location and shape data that was previously stored in the super pixels and now the constellations, estimation can be formed on the distance of a vertically classified constellation from the horizon line, and how tall it will be. Next during an iterative loop along the ground and vertical border, best fit polyline segments (VRML) are drawn. They are based on the previously calculated location data that now specifies where each object should be relative to the ground plane. Afterwards vertical fold lines are drawn from the polyline segments to the vertical--sky boundary. A model can then be "popped up" by folding at the polyline segments, and along each of the vertical lines. Finally the top of the vertical constellation is cut free from the sky along the vertical—sky border and a new 3D model is almost ready.

To finally generate the VRML version of the model, Photo Popup decides which direction to fold the vertical line segments on based on constellation data previously calculated. In essence it does logical comparisons of the constellation variables to see which areas are more likely to be folding in a particular direction together when looking at the model from the default camera view.

ICP and *k*d-Trees

Iterative Closest Point (ICP) is a method for aligning a set of three dimensional points M and D given an initial guess as presented in [1]. This is done by minimizing the sum of square of the Euclidean distance between the points. The result will be a rotation matrix R and a translation matrix T such that: $D' = DR + T$ The basic outline of this method is as follows:

1. Associate each point with its nearest neighbor
2. Estimate a transformation (R and T) that minimizes the cost function $C = \sum (P_i - P_j)^2$
3. Iterate again applying estimated R and T to D

This continues for as long as $C > \epsilon$ or until we have reached a specified number of iterations. For our application we used 60 iterations. We found that this was usually sufficient if the meshes were closely aligned or there was a strong correlation between the points in the two data sets. If the meshes do not align it is easy to simply run the algorithm again until we are satisfied with the results.

The implementation of ICP used in this application is an adaptation of a C++/Matlab implementation found at [2]. This code was adapted from its original form into a completely C++ implementation for use in our application. The calculation of the nearest neighbor can be time consuming for a large number of points. The naive implementation is as follows:

```
For all points in M:
  Distance = 0
  For all points in D:
    New distance =      Distance
between $M_i$      and $D_j$ squared
    If new distance <
  distance, set         $D_j$ as
closest         point
```

As you can see this is not very efficient at $O(N^2)$. A better approach is to minimize the set of points that need be compared. This can be done with a hierarchical structure such as a *k*d-tree. A *k*d-tree is a method for organizing points in k dimensional space into a hierarchical tree structure by splitting the data into two subsets. These subsets are created by placing a plane though the space that is orthogonal to the coordinate system. Through recursion on each subset a hierarchy of the data points is created. This allows for a quick traversal of the resulting tree structure to find eligible closest points.

The implementation of ICP from [2] included an implementation of *k*d-trees. Unfortunately the implementation was again tied to Matlab and written in structure based C style code. To allow for better integration with our application this code was ported to a more friendly C++ class based implementation.

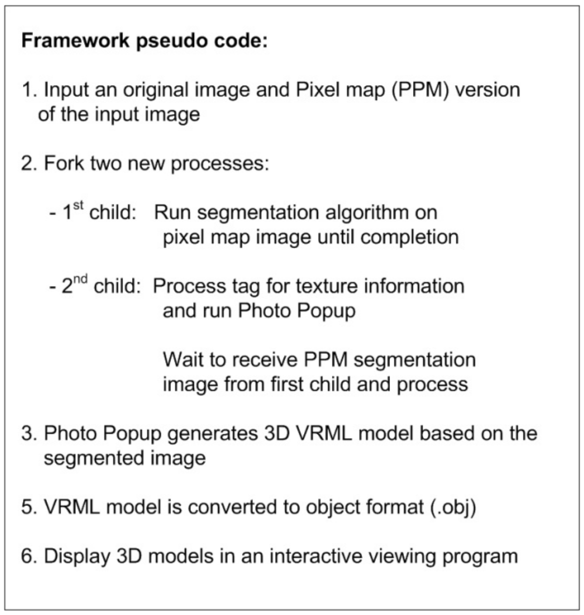
Software Implementation

The main contribution of this paper is a software application that enables users to create three dimensional models from their digital photographs and provides interactive tools for aligning these models to re-create scenes. This software streamlines and extends an existing implementation of Hoiem's and Efros Automatic Photo Pop-up work to support building three-dimensional models from multiple two dimensional photographs. When starting "Photo & Stitch" the user is prompted to give an image name and a pixel map version of the image. A pixel map is a simple save as command in most photo editing software, unlike making a segmentation map. After the input is received the user just has to wait until the OpenGL window is opened, where they will see the newly generated models. This process is much more user friendly and does not require any amazing math skills.

By developing an interactive user-interface and framework to automate the steps required to generate a single three-dimensional model from a photo this software is inspired by the work of [?] permitting manual alignment of models and includes an iterative closest point algorithm to refine user alignment when overlapping model regions are nearby.

Framework

This software is based on a framework that has been designed to provide a single screen Windows based application for aligning 3D models from generated from 2D photos. This framework was developed using the Microsoft Foundation Class (MFC) and Hoiem's and Efros Automatic Photo Pop-up software. MFC provides an application programmer interface (API) that supports the creation of user-interfaces, integration with OpenGL and calls to command line arguments. The process of generating 3D models from 2D photos using this framework is outlined in the following flowchart:



Framework pseudo code:

```
1. Input an original image and Pixel map (PPM) version
   of the input image

2. Fork two new processes:

   - 1st child:   Run segmentation algorithm on
                  pixel map image until completion

   - 2nd child:   Process tag for texture information
                  and run Photo Popup

                  Wait to receive PPM segmentation
                  image from first child and process

3. Photo Popup generates 3D VRML model based on the
   segmented image

5. VRML model is converted to object format (.obj)

6. Display 3D models in an interactive viewing program
```

Interactive Modeling Program:

This paper presents an interactive modeling software tool construct 3D scenes from multiple photographs by aligning 3D models produced generated Photo Popup. This process is described as follows 1) model loading 2) model alignment 3) texture alignment 4) model export.

1) Loading Models

The interactive modeling program displays photos user-selected photos as 3D models generated using Photo Popup. Each 3D model is represented as a VRML formatted mesh that is shown with the original input image as a texture. Before being loaded each VRML mesh is converted to a common object format (.obj).

2) Models alignment

Multiple 3D models can be aligned to create large 3D scenes. Two alignment modes are supported. The first, manual keyboard input using the arrow and page-up and down keys permits users to move objects along the X,Y and Z directions and select models respectively. For rotations both the shift key and arrows are pressed to orient models. The second method is to activate an automatic alignment using an iterative closest point algorithm. By selecting two models and pressing the ‘i’ key users can refine the initial alignment of two respective models.

3) Texture

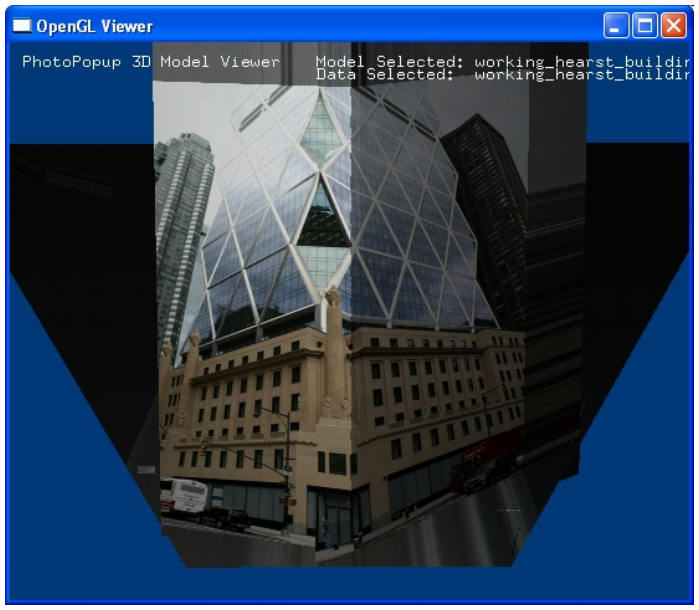
Each model receives a texture of the original photo. The alignment of this texture is based on the equation of a plane and is set to the x-axis by default. By pressing the w,a,s and d keys and -, = keys users can change the location and scale of a model texture respectively to better line up with model features.

4) Model export

Multiple models can be exported as a 3D scene. Following model and texture alignment users can export models as a standard (.obj) file by pressing the ‘e’ key.

Results

The results of this paper present a practical software application for creating 3D scenes from 2D photographs. Based on our research, the software we present is the first to extend Photo Popup research to support generating 3D scenes from multiple 3D Photo Popup models. This software is not perfect. It requires manual user intervention to provide an initial alignment of the images. This could be eliminated by providing a better, more robust alignment algorithm. Along this line, the ICP algorithm does not always perform well. If there is a low correlation between the two meshes it may or may not provide good results. In addition Photo Popup has inherit limitations that could not be avoided. For example only images with a visible sky, ground and horizon line are candidates for producing a well-defined model. Further more textures on the ground plane are often distorted when projected onto faces that are nearly orthogonal to the projection plane.


OpenGL Viewer
PhotoPopup 3D Model Viewer
Model Selected: working_hearst_buildir
Data Selected: working_hearst_buildir

Conclusion

In this paper a software application is presented to simplify the process of building three-dimensional scenes from multiple two dimensional photographs. Photo Popup is used to transform 2D photos to 3D meshes. By providing users with both a manual and automated method for aligning models we present a practical tool for creating 3D scenes from collections of photographs and enables users to create three dimensional models from collections of digital photographs of real-world scenes.

References:

[1] P. J. Besl, “A Method for Registration of 3-D Shapes”, IEEE Transactions on Pattern Analysis And Machine Intelligence, VOL. 14, NO. 2, February 1992

[2] P. Bergström “Iterative Closest Point Method, C++”, Matlab Central
http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=16766&objectType=file

[3]Image-Based Modeling and Photo Editing Byong Mok Oh Max Chen Julie Dorsey Fr´edo Durand
Laboratory for Computer Science
Massachusetts Institute of Technology Sigggraph 2001

[4] Automatic Photo Pop-up
Derek Hoiem Alexei A. Efros Martial Hebert
Carnegie Mellon University
Siggraph 2005

[5]Photo Tourism: Exploring Photo Collections in 3D
Noah Snavely
& Steven M. Seitz
University of Washington
Richard Szeliski
Microsoft Research

[6] Normalized Cuts and Image Segmentation
Jianbo Shi and Jitendra Malik, Member, IEEE University of Berkeley California

[7]S. Rusinkiewicz & M. Levoy, "Efficient Variants of The ICP Algorithm", Stanford University, 2001