CS268 Homework 4

Reconstruction, Manipulation, and Hallucination

November 17, 2015

Overview

The goal of this homework is to build and manipulate three dimensional representations of an observed scene and to develop a basic image formation pipeline. You will reconstruct and augment a 3D model of a scene using RGBD imagery (a common datasource for modern Computer Vision), and then generate a smooth video of your augmented scene reconstruction as seen from novel viewpoints. This assignment will involve assembling a system using many of the basic concepts you have covered in the course along with some basic concepts from Graphics, and be applicable to augmented/virtual reality and other popular modern applications of Computer Vision. Fig. 1 gives a visual overview of what to expect from this assignment, and a video demonstration has been included alongside this document. This being the final assignment, it is somewhat more experimental in nature with the onus on you to find ways to manipulate your reconstructed scene and measurements in order to generate interesting and/or realistic videos.
The *official* due date for this assignment is the last day of classes, **Dec. 4th**. To give you more flexibility in scheduling your time, you may submit the assignment anytime before midnight on the last day of the quarter, **Dec. 11th**, with no late penalty.
Please direct questions about the assignment to TA #4, Konstantine Tsotsos (`ktsotsos@ucla.edu`).

Implementation Notes

As with previous assignments, you may develop the implementation in whichever language you wish (note, assembly is not recommended). To reduce the implementation cost of this assignment, we have provided you with skeleton code implemented in Matlab that assembles much of the data processing pipeline for the assignment. You primarily should be 'filling in the blanks' of functional code that is missing. Since many of you have implemented homography and fundamental matrix estimation (from Homework 2 and 3) in other languages, we have also included (hopefully) working implementations of both of these modules in the code so that you do not need to re-implement them if you decide you need them in this assignment. The code we have provided relies on both VLFeat (`http://www.vlfeat.org/`) and VLG (`http://vision.ucla.edu/vlg/`), so please ensure you download and set these up properly.

- **Data:** We have collected a set of 27 RGBD images (4-channel images of Red-Green-Blue-Depth data) of a desk containing an assortment of typical household objects (typical in our houses, at least). These are split into two files in the Data directory, one containing RGB images and one containing depth data. Please see the text files in this directory for notes on how to interpret the data and for the calibration of the camera. You will also find several additional images that you may find useful for the assignment. If you have any problems reading the data, please let us know as soon as possible.
- **Missing Code:** You should be able to run the script hw4.m and generate a blank result. This is because several functional portions of the script are missing and have been labelled as regions you

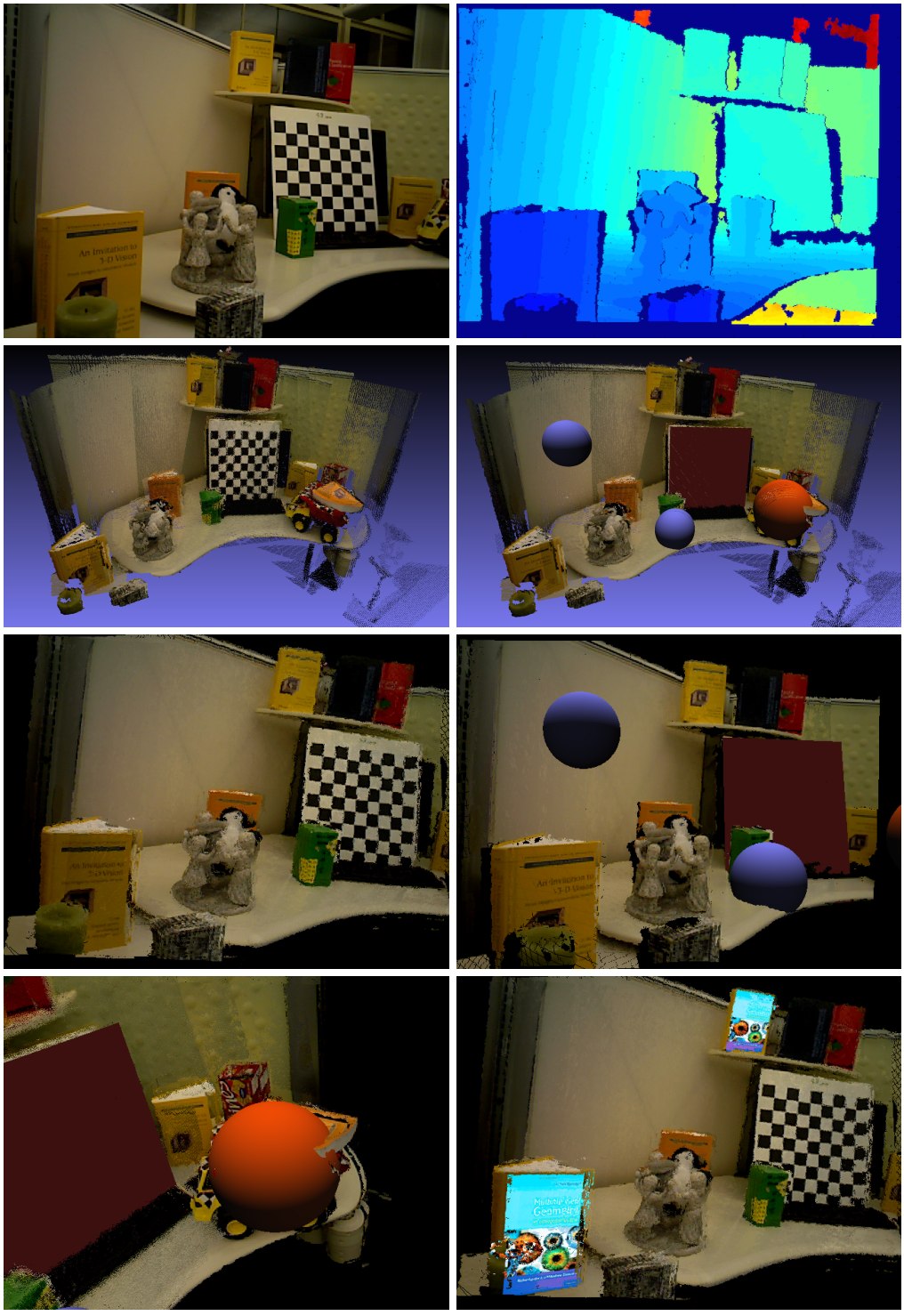

Figure 1: Overview of the results you should obtain for this assignment. From top to bottom and left to right, figures show: RGB image of scene, depth map of scene, 3D visualization of reconstructed scene, 3D visualization of scene with Lambertian objects inserted, re-projection of reconstructed scene, re-projection of manipulated scene, alternate re-projection of manipulated scene, re-projection of scene manipulated to replace instances of the class textbook with a blue book.

should implement. Also, several helper functions have been left blank and will require your attention. These are: ProjectPoint.m, UnProjectPoint.m, and insertLambertianSphere/Plane (if you choose to use them). Please let us know of any issues with the code provided as soon as possible.

1 Reconstruction

Your first goal is to build a model of the scene that generated the images we have provided for you. This model should encode the aspects of the geometry and photometry that you would need in order to re-create the images using the image formation models you have covered in class. This part of the assignment consists of several tasks:

1. Select a coordinate frame for your model: The coordinates of your 3D model of the scene must be expressed with respect to some reference or origin frame. You will have to align all of the 3D data from the RGBD images to this coordinate frame in order to build your model. The camera frame of one of the RGBD images is typically a good choice for this.

2. Align RGBD images: Each RGBD image we have provided you encodes and color image and the depth associated to each pixel. From each one of these, you are able to build a set of 3D points and associated RGB intensities in the frame of that camera. It is up to you to determine the rigid transformations that align the 3D points of each image to the coordinate frame your have chosen as the origin of your model. By align, we mean determining the relative rotation and translation between each camera and your chosen reference frame, and using these transformations to express all of the 3D points in that reference frame. You are free to use techniques discussed in class, or to explore other options outside of the course material.

3. Visualize your colored 3D reconstruction: Now that you've aligned the RGBD frames of the scene into a single coordinate frame, you should visualize it and inspect your handiwork. The visualization shown in the second row of Fig. 1 is similar to what you should obtain. Does your reconstruction look correct? What kinds of problems do you see? How would you go about solving these problems?

4. Optional extensions for extra credit:

   - **Bundle Adjustment:** Is the alignment of the point clouds from each RGBD frame perfect? Nonlinear optimization techniques (briefly discussed in lectures) such as Bundle Adjustment could be used to significantly improve the final point cloud. You may explore external implementations of some of these techniques and compare the resulting reconstructions to your own method. Alternatively, you may write and solve your own objective function that directly optimizes a photo-consistency measure in order to optimize over points and transformations. Detail any experimentation/formalism used here in your report.
   - **Point Cloud Reduction:** The final point cloud is extremely dense and redundant. Can you think of any clever and efficient ways to reduce the number of points in the cloud that do not impact your results? Document any experimentation in your report.
   - **Aesthetics:** You may have a good deal of outliers, particularly if you have any misalignments in your aligned pointclouds. Can you think of any clever algorithms to fill in segments that are missing data or otherwise unsightly?
   - **Representation:** The representation you have built by aligning several RGBD frames is just a cloud of dots in 3D with colors attached to them. While, at the atomic level, this may be a faithful reproduction of reality, we do not interact with the world at that level of abstraction. We interact with a world of *surfaces*. Can you explore any way of 'upgrading' your cloud of dots to a denser, or surface-based representation? How would this change your pipeline for rendering new views of the scene? Can this improve the quality of your reconstruction? Detail any additional experimentation/formalism in your report

2 Manipulation

In order to explore the ideas of augmentation you will manipulate your reconstructed scene model to make it more exciting and interesting to look at. For simplicity, you should use the Lambertian reflectance model for any objects you add to your scene. The goal is to experiment with manipulating the model you have created and to work through the use of a simple reflectance model. As this is part of the open-ended portion of the assignment, you have several options for how to manipulate your scene model.

Choose your own adventure!

1. **Insertion of simple Lambertian objects:** Your task will be to insert several simple geometrical objects into the scene in semantically relevant ways. For instance, you may cover some of the small objects on the table with spheres and replace some planar surfaces in the scene with planes. You should also experiment with multiple light sources and different albedo combinations and document your findings and any interesting results. Some code prompts for these have been provided.
   Difficulty: Easy.
   Implementation Complexity: Low.
   Potential for interesting resulting video: Low.

2. **Insertion of complex Lambertian objects:** Your task will be to insert several complex objects into the scene. For instance, you may use the sphere generation code, deforming them to generate ovoid objects, or generate polyhedral objects to add to the miscellaneous objects on the table top using the plane generation code. You could also add CAD models from online 3D model repositories to the scene.
   Difficulty: Medium Easy.
   Implementation Complexity: Medium.
   Potential for interesting resulting video: Medium.

3. **Replacement of books using a textured Lambertian surface:** In Homework 2 you developed a pipeline to find homographies between a pair of images. In this scene, there are several planar objects for which you have a template image (the course textbook). With a simple extension of your pipeline, you can detect all instances of the textbook in the scene and replace the cover with with a different surface modeled as Lambertian. For example, you could replace them with the image of the *other* Computer Vision textbook provided and determine appropriate lighting conditions so that they appear to be a part of the original scene.
   Difficulty: Medium.
   Implementation Complexity: Medium High.
   Potential for interesting resulting video: High.

4. **Insertion of deforming/moving Lambertian objects:** Your task will be to insert at least one deforming or moving object into the scene. For instance, you could insert an ovoid object that deforms through the video sequence, a simulated lava lamp, a ball rolling across the surface of the table, or an articulated walking transformer made of planar segments. Note: Objects that move or deform through the sequence should obey basic laws of physics and collide with other objects in the scene (i.e. no clipping).
   Difficulty: Hard.
   Implementation Complexity: High.
   Potential for interesting resulting video: Very High.

5. **Non-Lambertian objects:** You may upgrade any of the options above to use a non-Lambertian reflectance function if you wish. This would likely lead to much more interesting results.

6. **Other:** Up to you, but should be more interesting than the simplest option above. Have fun!

3 Hallucination

Now that we have a reconstructed model and a reflectance function, we can use these along with the image formation model to hallucinate new images of the scene that we did not record. You will augment the trajectory of the camera and generate a temporally smooth video sequence of your reconstructed scene. Some code has been provided to you for rendering an image from a particular vantage point, but determining the vantage points you need to generate an interesting and aesthetically pleasing video is up to you.

Choose your own adventure!

1. **Temporal up-sampling through interpolation:** The original images do not form a smooth video, as they were just a collection of different viewpoints. Your task will be to interpolate between these disparate viewpoints in order to generate a sufficiently smooth and believable video sequence. Your result should should have sufficient temporal continuity that is nearly indistinguishable from a continuous video taken while looking at the reconstructed scene.
   Difficulty: Easy.
   Implementation Complexity: Low.
   Potential for interesting resulting video: Medium.

2. **Novel camera trajectory:** Your task will be to generate a completely novel, smooth trajectory for the camera and use it to generate a continuous video sequence of the scene. Your trajectory should be at least comparable in length and generality of motion to the original set of viewpoints you used to reconstruct your scene. You may also want to use the control over your trajectory to highlight interesting views of your manipulations of the scene.
   Difficulty: Medium.
   Implementation Complexity: Medium.
   Potential for interesting resulting video: High.

3. **Other:** Up to you, but should be more interesting than the simplest option above. Have fun!

4 Final Product

Your final product should be at least one video and a report. If you have problems using the entire image sequence to build your scene model, a small subset (for example, not < 3) should suffice. Your report and video/image sequences should contain evidence and discussion of any experimentation or any interesting results. The report should also detail all necessary formalism or algorithms used in your final system. Several guiding questions for things to discuss in your report are as follows:

- The Lambertian reflectance model: What is it? What are its major assumptions and limitations? What would make a reflectance model non-Lambertian? How are you using it in your system?
- Aligning RGBD Images: How did you align the 3D points from the RGBD images? What coordinate frames are involved?
- Planes, spheres, and other objects: How did you generate these? How did you assign intensities to them in the final images? If they deform/move/interact with the scene how did you accomplish this?
- Viewpoint interpolation and smooth trajectories: How did you interpolate between viewpoints? What does this actually mean? How did you generate a smooth trajectory? What does a temporally smooth video sequence mean? Under what conditions can you say that it is indistinguishable from a continuous movie? Are there any key mathematical notes you need to discuss here?
- Problems: What problems remain in the system pipeline that lead to poor quality results? When does your system fail?