Generating High Quality Mobile Image Datasets for the Evaluation of Computer Vision Algorithms

Middlebury College Department of Computer Science

Summer 2019

Support through NSF Grant IIS–1718376 is gratefully acknowledged

Tiansheng Sun
Camera calibration, realistic trajectory recording

Toby Weed
Control program, structured lighting, image processing

Guanghan Pan
Robot control, realistic trajectory recording, web scripts

Daniel Scharstein
Charles A. Dana Professor of Computer Science

Abstract

This research project aims to provide a system for capturing datasets to be used in the evaluation and testing of dense stereo and optical flow computer vision algorithms. The collected datasets will be constituent of high-resolution ambient images captured by smartphones and ground-truth depth maps generated by a structured lighting system. They will be published as the next generation of the well-known Middlebury Stereo Vision benchmarks, and are intended to advance stereo and optical flow vision research by supplying highly accurate ground truth depth maps of static indoor scenes.

The main MobileLighting control program is written in Swift and C++ for macOS. It coordinates the different devices used for scene capture, including numerous projectors used to cast structured lighting onto the scene, a robot arm which moves the smartphone, and an iOS app (also written in Swift) which captures photos and IMU data.

To enable repeatable photo and video capture from highly customizable viewpoints and trajectories, the smartphone is mounted on the head of a UR5 robot arm. The arm aims to reproduce actual human arm trajectories recorded with SteamVR using an HTC VIVE tracker. After capture, the realistic kinetic data is smoothed and numerous reference points are chosen along the focal point's path. These trajectories are then loaded onto a Rosvita control server, which mediates communication between the mac control program and the robot.

Once the structured light images and ambient data have been collected in coordination with the projectors, robot, and smartphone, the mac program processes the image data to produce precise depth maps of the scene. In order to make the ground truth data as accurate as possible, the program produces camera calibration matrices to correct warping from the capture device's lens and differences in the relative angle of each viewpoint. These matrices are generated by detecting, in calibration images taken by the system, a custom version of AruCo barcode patterns printed in documented intervals on flat boards.

Motivation

Research in multiview stereo vision and optical flow vision is held back by the absence of precise methods for evaluating the accuracy of different algorithms.

This project will fill that void by providing diverse datasets with subpixel accurate ground truth depth maps which researchers will be able to compare the results of their algorithms against.

The input (ambient) images provided will be true to modern mobile use cases, with images captured by mobile phones under a variety of different lighting conditions.

Calibration

The first step of the process involves taking images with specific coded patterns for intrinsic and extrinsic calibration, which calculates the intrinsic and extrinsic parameters of the camera model we are using.

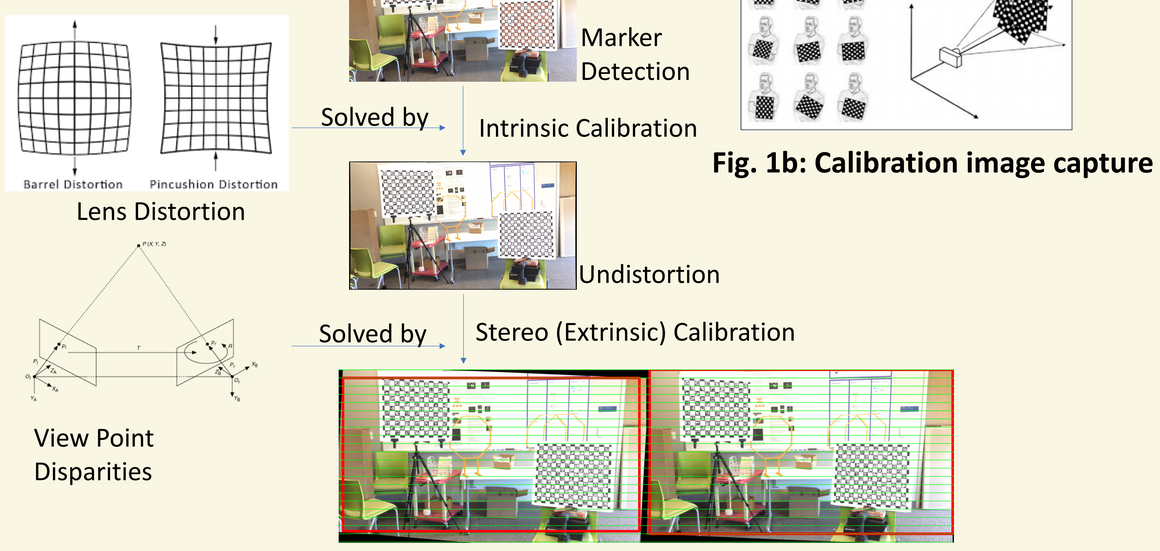
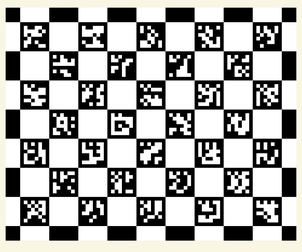


Fig. 1a: Calibration process

Fig. 1b: Calibration image capture



Calibration Boards
For calibration purposes, we use our customized encoded ArUco boards. ArUco marker is a wide square marker with black corner and an inner binary matrix which determines its unique ID, whose correspondence can be used for calibration purposes. It works successfully with partially occluded views. We have also experienced with ChArUco and Chessboard patterns.

Ground Truth

Ground Truth Calibration Problem
One of the major components required of the finished datasets is highly accurate ground truth depth information. There are a number of highly accurate systems available for 3D scanning. However, calibrating the output of any of these systems for comparison with the output of algorithms run on images from a camera can be daunting, especially when accounting for the relative location of depth sensors versus cameras.

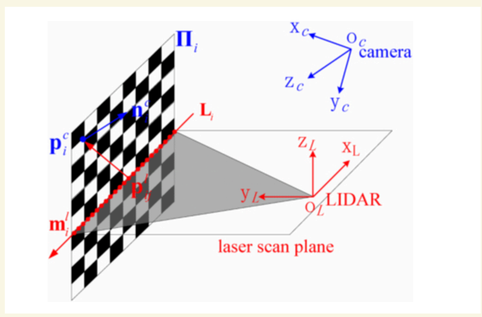


Fig. 2: example of extrinsic calibration between camera and depth scanner that we avoid

Structured Lighting
To avoid the ground truth calibration problem, this system uses the same sensor (the smartphone camera) to obtain both ground truth depth information and stereo vision input images. In order to acquire ground truth, the program uses a "structured lighting" system which projects binary patterns onto the scene to get a unique pixel code for each point. The pixel codes are then compared between views; the more a pixel moves, the closer the surface is to the camera.

Fig. 3: projecting structured lighting onto a scene

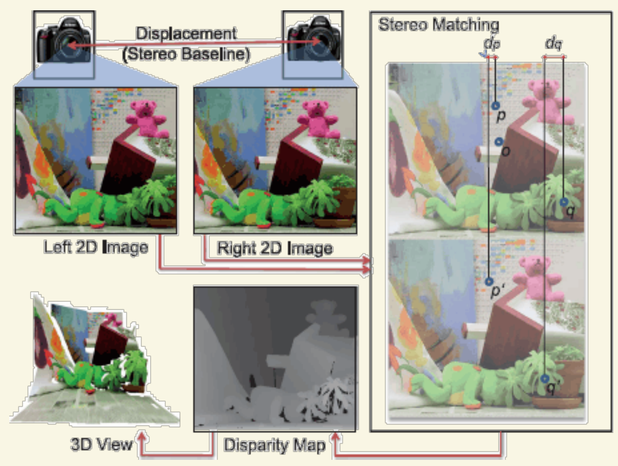


Fig. 4: comparing 2D codes of points on image to extract depth information

Control and Coordination

The program is controlled from the central command-line application, MobileLighting Mac. MobileLighting Mac is responsible for commanding the smartphone for image capture, projecting structured lighting from the projectors (attached via a switcher box), commanding the UR5 robot via the Rosvita server, running calibration, processing the images, and recording all relevant scene data. MobileLighting Mac connects to MobileLighting iPhone and MLRobotControl wirelessly using network sockets.

Image capture occurs on a device running an iOS app, MobileLighting iPhone, which takes photos under various exposures and does some image processing automatically.

The movement of robot arm is controlled by the socket server, MLRobotControl, which is responsible for moving robot to specified viewpoint, executing pre-defined trajectories and sending the status of robot back to MobileLighting Mac.

MobileLighting Mac

- Command line tool
- Coordinates between different parts of the system, does image processing and calibration, records data.
- Written in Swift, C++, and Objective-C using Xcode

MobileLighting iPhone

- iOS application
- Responsible for image capture and some image processing.
- Written in Swift using Xcode

MLRobotControl

- Socket server
- Coordinates between the communication of robot and MobileLighting Mac, sends commands to robot arm
- Written in Python using Rosvita

Viewpoint Management with UR5 Robot Arm

In order to have repeatable photo capture from different viewpoints, the smartphone is mounted on the tool head of a UR5 robot arm. Our main method of communicating with the robot arm is through Rosvita, a third-party robot programming environment based on ROS (Robot Operating System)

The robot arm can be instructed to move to a certain position in two different ways: joint positions and poses.

Joint Positions:
Since the robot arm has six joints, a position can be specified using an array of six angles, each represents the angle of a joint. This array gives a unique configuration of the robot arm. As a result, one option to define the viewpoints for the robot arm is to manually record a fixed number of joint positions and have the robot arm to move through them smoothly.

Poses:
The pose, on the other hand, contains an array with seven numbers. The first three numbers are the cartesian coordinate of the robot's tool head. The next four are the quaternions representing the orientations and rotations of the tool head in three dimensions. Therefore, another more advanced option is to extract the human motion poses data using SteamVR with HTC VIVE tracker, then have the robot arm to mimic human motion as accurate as possible by passing the recorded data to Rosvita. The viewpoints will then be selected from the recorded trajectory.

Fig. 5: Smartphone mounting to the tool head

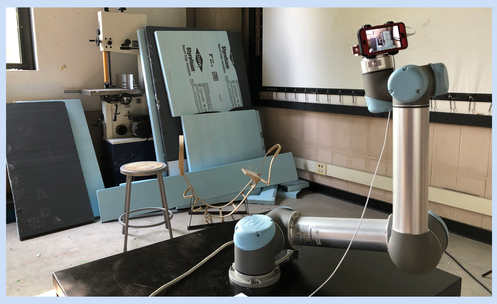


Fig. 6: Robot arm moving to different viewpoints during scene capture

VIVE Realistic Human Motion Tracking

To imitate the human arm motion and then execute the trajectory on the robot arm, VIVE tracker is used along with a base station to extract realistic pose information, which will then be passed to the UR5 robot arm through Rosvita. An image containing the information of the recorded trajectory will also be generated after saving the data. In Rosvita, the viewpoints will be selected according to the tracked poses.

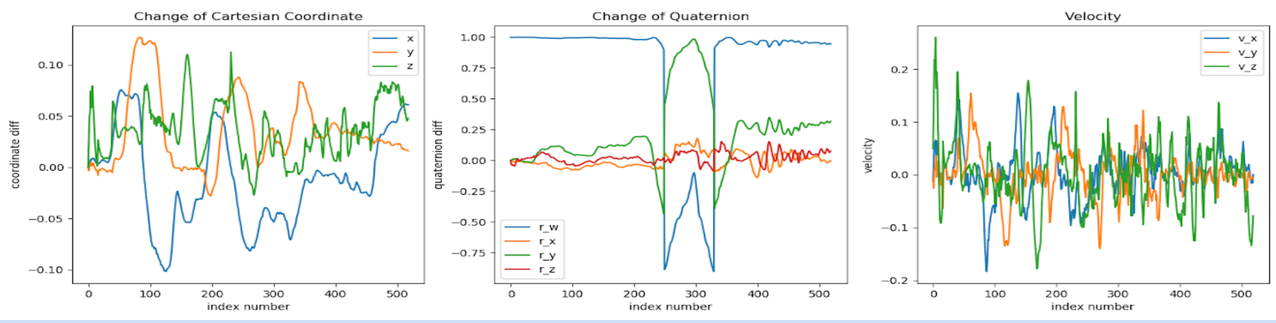


Fig. 7: The image generated that contains graphs showing the information of the recorded change of Cartesian Coordinates, change of quaternion, and velocity through time.

Fig. 8: the base station

Fig. 9: the motion tracker and sensor

Image Processing

To get ground truth depth maps of the highest possible quality, we use an image processing pipeline consisting of 6 steps:

Decode images
Decode the structured lighting images to get a unique pixel code for each point in the scene. This consists in projecting ever narrower lines in both the x- and y- directions (see fig. 3).

Rectify images
Using the extrinsic calibration parameters generated from calibration, rectify images from adjacent camera views so that they appear to lie on the same plane.

Disparity match images
Merge the x- and y- decoded images to get a unique pixel code for each point. Then compare the pixel codes from two views to get a disparity map.

Merge disparity maps
Merge disparity maps produced by different projectors to fill in as many depth values as possible as accurately as possible.

Reproject merged disparity maps
The merged disparities are used to self-calibrate the projector positions. Once the projector relationships are known, half-occluded regions can be filled in with disparity values.

Merge2
Put together original disparities, merged disparities, and reprojected disparities to fill in all possible values.

These steps are all implemented using C++ in MobileLighting Mac.

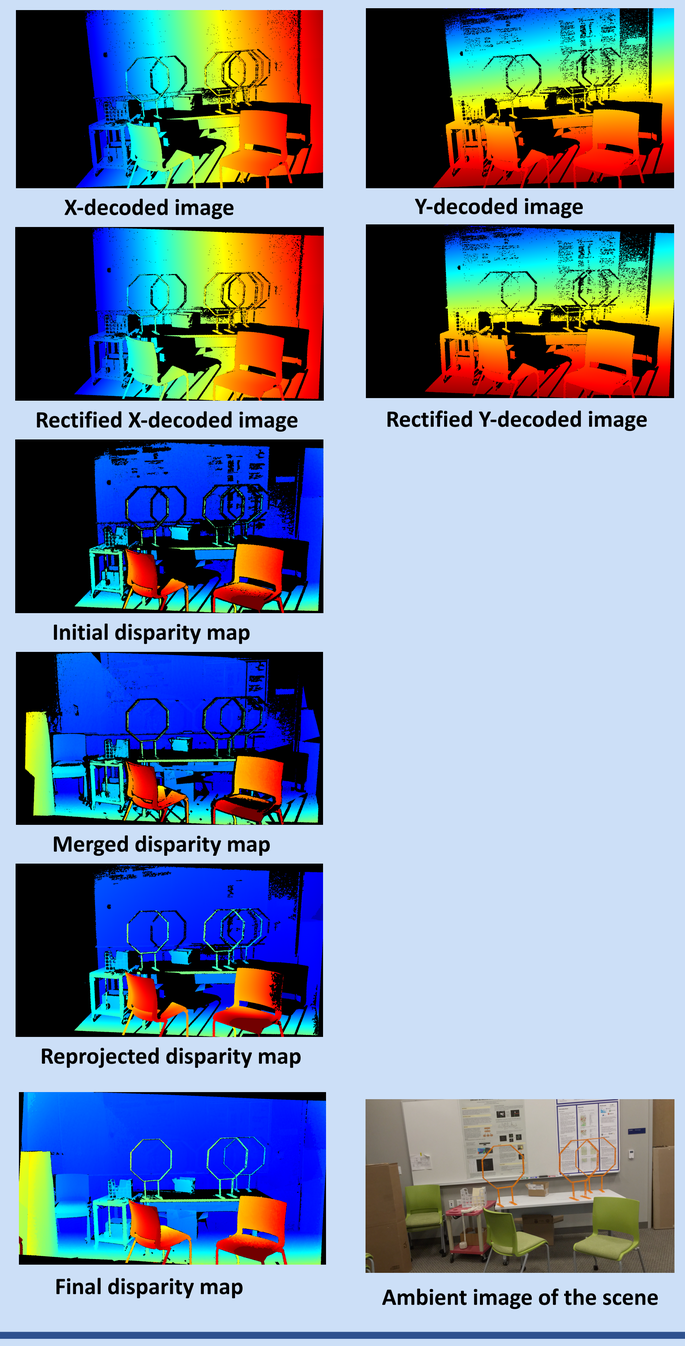


Fig. 10: the image processing pipeline

References

Fig 1b from http://apps.man.poznan.pl/trac/stereovision
Fig 2 from https://iopscience.iop.org/article/10.1088/0957-0233/25/6/065107
Fig 4 from https://link.springer.com/chapter/10.1007/978-1-4471-5520-1_6

Thanks to: Eamon McMahon for his help in mounting new ChArUco boards and Rick James for our WiFi routers.