SOFTWARE ENGINEERING

PROJECT REPORT

3D Scanner

November 27, 2016

Andrey Sakryukin
Hassan Saeed
Islam Mokhtari
Ivan Mikhailov
Katherine Sheran
Minh Vu
Saed Khwaldeh
Solene Guillaume
Tajwar Aleef
Usama Pervais
Yeman Hagos

Contents

Abstract

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Overview

1.2 Objectives

1.3 Structure of Report

Chapter 2

Background

2.1 KINECT

2.2 3D RECONSTRUCT

Chapter 3

Related Tools and Libraries

3.1 Git

Introduction

Why use GIT?

Git is a mature, actively maintained open source project originally developed in 2005 by Linus Torvalds, the famous creator of the Linux operating system kernel. A staggering number of software projects rely on Git for version control, including commercial projects as well as open source. Developers who have worked with Git are well represented in the pool of available software development talent and it works well on a wide range of operating systems and Integrated Development Environments (IDEs).

Having a distributed architecture, Git is an example of a Distributed Version Control System (DVCS). Rather than have only one single place for the full version history of the software as is common in once-popular version control systems like Version Control System (VCS) or Subversion (SVN), in Git, every developer's working copy of the code is also a repository that can contain the full history of all changes.

In addition to being distributed, Git has been designed with performance, security and flexibility in mind.

The raw performance characteristics of Git are very strong when compared to many alternatives. Committing new changes, branching, merging and comparing past versions are all optimized for performance. The algorithms implemented inside Git take advantage of deep knowledge about common attributes of real source code file trees, how they are usually modified over time and what the access patterns are.

Git has been designed with the integrity of managed source code as a top priority. The content of the files as well as the true relationships between files and directories, versions, tags and commits, all of these objects in the Git repository are secured with a cryptographically secure hashing algorithm called SHA1. This protects the code and the change history against both accidental and malicious change and ensures that the history is fully traceable.

One of Git's key design objectives is flexibility. Git is flexible in several respects: in support for various kinds of nonlinear development workflows, in its efficiency in both small and large projects and in its compatibility with many existing systems and protocols.

3.2 QT FRAMEWORK AND QT CREATOR

Why we use QT but not Microsoft Visual Studio??

8 of Top 10 Fortune 500 use Qt Qt is the software development framework of choice by engineers in over 70 industries worldwide for creating, building and deploying millions of connected embedded devices and applications. They chose Qt because it's intuitive, cross-platform and saves time. When you develop with Qt, you create more, code less and deploy everywhere — and ahead of the rest.

Qt is a cross-platform application framework that is widely used for developing application software that can be run on various software and hardware platforms with little or no change in the underlying codebase, while still being a native application with native capabilities and speed. Qt is currently being developed both by The Qt Company, a company listed on the Nasdaq Helsinki Stock Exchange and the Qt Project under open-source governance, involving individual developers and firms working to advance Qt. Qt is available with both commercial and open source GPL 2.0, GPL 3.0, and LGPL 3.0 licenses.

Qt Creator provides a cross-platform, complete IDE for application developers to create applications for multiple desktop, embedded, and mobile device platforms, such as Android and iOS. It is available for Linux, macOS and Windows operating systems. For more information, see Supported Platforms.

3.3 MICROSOFT SDK

Microsoft Windows Software Development Kit (SDK), and its predecessors Platform SDK, and .NET Framework SDK, are SDKs from Microsoft that contain documentation, header files, libraries, samples and tools required to develop applications for Microsoft Windows and .NET Framework. Platform SDK specializes in developing applications for Windows 2000, XP and Windows Server 2003. .NET Framework SDK is dedicated to developing applications for .NET Framework 1.1 and .NET Framework 2.0. Windows SDK is the successor of the two and supports developing applications for Windows XP and later, as well as .NET Framework 3.0 and later.

Windows SDK allows the user to specify the components to be installed and where to install them. It integrates with Visual Studio, so that multiple copies of the components that both have are not installed; however, there are compatibility caveats if either of the two is not from the same era.[4][5]

Information shown can be filtered by content, such as showing only new Windows Vista content, only .NET Framework content, or showing content for a specific language or technology.

Use the latest sensor and the Kinect for Windows SDK 2.0 to create Windows apps that run on PCs and tablets. Be sure to select "Windows 8.1" as a supported target during application authoring to ensure that your application also runs on Windows 10. Kinect for Windows v2 apps will continue to work after an upgrade to Windows 10. As always, we recommend that that you test applications on new platforms as they are released.

3.4 OpenNI2

3.5 Point Cloud Library

3.6 Kinect Fusion

Kinect Fusion provides 3D object scanning and model creation using a Kinect for Windows sensor. The user can paint a scene with the Kinect camera and simultaneously see, and interact with, a detailed 3D model of the scene. Kinect Fusion can be run at interactive rates on supported Graphics Processing Units (GPUs), and can run at non-interactive rates on a variety of hardware. Running at non-interactive rates may allow larger volume reconstructions.

Chapter 4

Implementation

4.1 Project Management

4.2 Coding Standard

4.3 Setup

4.4 Graphical User Interface

Chapter 5

Implementation

5.1 Surface Measurement

5.2 Sensor Pose Estimation

5.3 Global Model Update

5.4 Surface Extraction

5.5 Post-processing

Chapter 6

Result and Discussion

Chapter 7

Conclusion

Bibliography