Into the ROS

Advanced ROS Network Introspection

Praxis der Softwarentwicklung
Sommersemester 2014

P f l i c h t e n h e f t

Auftraggeber

KIT - Karlsruher Institut für Technologie
Fakultät für Informatik
Institut für Anthropromatik und Robotik (IAR)
Intelligente Prozessautomation und Robotik (IPR)

Betreuer: Andreas Bihlmaier
andreas.bihlmaier@gmx.net

Auftragnehmer

Name	E-Mail-Adresse
Alex Weber	alex.weber3@gmx.net
Matthias Hadlich	matthiashadlich@yahoo.de
Matthias Klatte	matthias.klatte@go4more.de
Micha Wetzel	micha.wetzel@student.kit.edu
Sebastian Kneipp	sebastian.kneipp@gmx.net

Karlsruhe, 27.05.2014

Inhaltsverzeichnis

1 Zielbestimmung

Die zu entwickelnde Softwarelösung soll sich als Überwachungsinstrument in die Robot Operating System (ROS) Middleware einfügen. Dabei soll überwacht werden, ob jeder Prozess, der im ROS-Netzwerk enthalten ist, definierten Sollwerten entsprechend ordnungsgemäß Daten übermittelt.
Gerade durch die gegebene Problematik eines auf mehrere Hosts verteilten Systems, bereitet das Erfassen und Verstehen von auftretenden Problemen zum aktuellen Stand des ROS Schwierigkeiten. Das vorliegende Projekt erfasst den Ist-Zustand der Prozesse im ROS-Netzwerk ohne Leistungseinbußen kontinuierlich und visualisiert diesen im Vergleich zu definierten Soll-Werten. Auftretende Fehler und Abweichungen der von den Spezifikationen werden den Benutzer deutlich angezeigt. Es lassen sich Gegenmaßnahmen für verschiedene Fehler definieren.

1.1 Musskriterien

- Minimale Erweiterung bestehender Knoten zur Übermittlung von Metadaten
- Publizierung der Metadaten mit defnierter Frequenz auf Topics
- Deaktivierung der Datenerfassung einzelner Knoten
- Definition von Soll-Zuständen
- Graphische Übersicht über Abweichungen von Soll- und Ist-Zustand der Knotendaten
- Statuserfassung der Host-Computer

1.2 Wunschkriterien

- Überwachung weiterer ROS Bestandteile: Service, Parameters
- Selbstkontrolle der Knoten
- Ist-Zustand als Soll-Zustand definieren
- Ergänzende Übersichten mit dem ROS Graphen und Plots über den zeitlichen Verlauf

1.3 Abgrenzungskriterien

- Das Projekt wird keine vollständige Automatisierung in Fehlerfällen umsetzen.
- Überwachung der Netzwerkinfrastruktur

2 Produkteinsatz

2.1 Anwendungsbereiche

Angewandt wird die Software zum fehlersuchen und überwachen von kompletten ROS Netzwerken.

2.2 Zielgruppen

Die Zielgruppe besteht aus Entwicklern und Administratoren von Umgebungen in denen ROS eingesetzt wird. Es wird technisches Verständnis der Nutzer vorausgesetzt.

2.3 Betriebsbedingungen

- Zur Fehlererkennung wird die Software gestartet um eine genauere Ursache des Fehlers festzustellen.
- Während des Produktivbetriebs wird die Überwachung zugeschaltet um Versagen in dem System zu erkennen und automatische Gegenmaßnahmen einzuleiten.

3 Produktumgebung

Produktumgebung selbe wie davor kp ob wir die subsections brauchen ich hab sie mal drinn gelassen

ja der name sagt ja auch schon alles

3.1 Software

3.1.1 Implementierung

- GNU/Linux (Ubuntu 12.04)
- ROS Hydro oder neuer
- Python 2.7 oder neuer
- QT 5.0 oder neuer

3.1.2 Dokumentation

- Latex
- Wiki im GitHub Repository

3.2 Hardware

..

3.3 Orgware

..

3.4 Produktschnittstellen

..

3.5 Lizenz

4 Funktionale Anforderungen

4.1 Gesamtsystem

4.1.1 Pflicht

/FA0100/ Dezentrale Erfassung von Metadaten: Anzahl Publisher und Subscriber, Bandbreite, Frequenz, Latenz, Jitter

/FA0200/ Monitoring Knoten zum zentralen Abgleich des Soll- und Ist-Zustandes

/FA0210/ Warnungen und Fehlernachrichten bei maßgeblichen Abweichungen

/FA0300/ Definition eines ROS Message Types für Metadaten

/FA0400/ Eigenständiger Knoten zur Überwachung der Hardware des Host-Systems (CPU Auslastung, CPU Temperatur, RAM, Festplatten-Speicher)

4.1.2 Optional

/FA0500/ Überwachung weiterer ROS Komponenten wie Services und Parameters

/FA0600/ Festlegen des Ist-Zustandes als Soll-Definition

/FA0610/ Korrelationen zwischen Empfangs- und Sendeverhalten eines Knotens erkennen

/FA0700/ Anpassung des Sendeverhaltens des Systems an Netzwerkgegebenheiten

/FA0800/ Definiton der Netzwerktopologie sowie darstellung der Auslastung physischer Verbundungen

/FA0900/ Integration mit roswtf

4.2 Soll-Spezifikation

4.2.1 Pflicht

/FA1000/ Parametriesierung für Topics, Hosts und Knoten getrennt

/FA1010/ Obere und untere Schranken für Metadaten definierbar

/FA1100/ Laden der Soll-Spezifikation einmalig bei Start des Überwachungsknotens

/FA1110/ Weitergabe der Soll-Spezifikation an den ROS Parameter Server bei Knotenstart

/FA1200/ Möglichkeit nur Teile des Netzwerkes zu überwachen

/FA1210/ Teilsysteme können sich überlappen

4.2.2 Optional

/FA1300/ Ein Monitoringknoten verwaltet alle Teilsysteme

4.3 API

4.3.1 Pflicht

/FA1400/ Die Metadaten werden durch Hinzufügen eines Funktionsaufrufes zu bestehenden Callbacks erfasst, alternativ durch eine geeignete Klasse, die von Subscriber erbt

/FA1500/ Die Metadaten werden auf einem Topic mit definierter Frequenz publiziert

/FA1600/ Die Metadatenerfassung lässt sich über Parameter deaktivieren

4.3.2 Optional

/FA1700/ Knoten sind in der Lage, sich anhand ihrer Soll-Metadaten selber zu überwachen

/FA1800/ Das System wird auch für C++ Knoten implementiert

4.4 GUI

4.4.1 Pflicht

/FA1900/ Die grafische Benutzeroberfläche bietet eine Visualisierung des Soll-Ist-Vergleichs von sowohl Knoten als auch Hostsystemen

4.4.2 Optional

/FA2000/ Der Soll-Ist-Vergleich wird in einer Visualisierung des ROS Graphen dargestellt

/FA2100/ Geeignete Metadaten werden im ROS Graphen dargestellt

/FA2200/ Es wird ein zeitlicher Verlauf der Metadaten aufgezeichnet und grafisch dargestellt

/FA2300/ Knoten werden im Graphen nach ihrem Host gruppiert angezeigt

/FA2400/ Knoten werden als Gruppen der unterschiedlichen Soll-Spezifikationen angezeigt

5 Nichtfunktionale Anforderungen

5.1 Produktleistungen *Pflicht*

/NF0100/

/NF0110/

/NF0200/

/NF0100/ Die GUI soll schnell starten und interaktiv bedienbar sein

/NF0200/ Möglichst kein Overhead im Release-Modus

5.2 Produktleistungen *Optional*

/NF0300/ Flexibler Umgang mit unterschiedlichen Bildschirm- und Bildauflösungen

/NF0400/ Integration in das OpenCV Test Framework

5.3 Qualitätsanforderungen *Pflicht*

/NF0500/ Keine signifikanten Speicherlecks

/NF0600/ Erweiterbarkeit um zusätzliche OpenCV Operationen und Visualisierungen

/NF0700/ Modularer Aufbau (API und GUI)

/NF0900/ Einhaltung der OpenCV und Qt Konventionen

/NF1000/ Ausführliche Dokumentation der API und des GUI

5.4 Qualitätsanforderungen *Optional*

/NF1100/ Dokumentation des internen Codes mit Werkzeug

/NF1200/ OpenCV geeigneter Aufbau des Build-Systems

/NF1300/ Abdeckung durch Tests

/NF1400/ Keine Resource-Leaks

/NF1500/ Threadsafety *Im C++11-Modus*

/NF1600/ Toleranz gegenüber fehlerhaften API-Aufrufen

/NF1700/ Kein undefiniertes Verhalten

6 Produktdaten

beschreibung der inhalte der metadaten

angabe des config konzepts

7 Systemmodell

das selbe wie im kapitel davor nicht sicher was wir da genau reinschreiben sollen

7.1 Szenarien

7.2 Anwenungsfälle

8 Benutzeroberfläche

hier kommen dann ein paar bilder und dazugehörige beschreibungenen unserer gui

- blasdgfs
- sdfg
- was man halt alles so allgemeines reinschreiben kann

8.1 Skizzen bzw. Bilder

8.1.1 name des bildes

Abbildung 8.1: bildunterschrift

Ein ganz ganz langer text zum bild
kdfnakgflkadshfkjsfkljsadfjköadshfölkdf
hsgkjöldhflkghsfdlkghödklfhglkdsjglödshgkdsfhögh
sdflkendgkjhdsfköghadfsopfghaoifhaigtfuiaerhgiuag

8.1.2 name des bildes

Abbildung 8.2: bildunterschrift

Ein ganz ganz langer text zum bild
kdfnakgflkadshfkjsfkljsadfjköadshfölkdf
hsgkjöldhflkghsfdlkghödklfhglkdsjglödshgkdsfhögh
sdflkgkjhdsfköghadfsopfghaoifhaigtfuiaerhgiuag

9 Qualitätsanforderungen

Wartbarkeit, Zuverlässigkeit, Erweiterbarkeit etc. anpreisen

10 Testfälle

11 Abgrenzungskriterien

wie der name schon sagt hier dann die abgrenzungskriterien

12 Glossar

des ist jetzt mal der glossar von letztem jahr den können wir ja ganz schnell bearbeiten und unsere sachen einfügen

12.1 Allgemein

API Application Programming Interface. Eine Schnittstelle (s. Interface), über welche andere Programme auf der Quelltextebene auf dahinter verborgene Funktionalität zugreifen können

Augmented Reality „Erweiterung der Realität“ durch einen Computer, etwa bei der Einblendung von Informationen in ein Bild der Umgebung, das auf einem Smartphone angezeigt wird.

Bug Fehlerhaftes Verhalten eines Programmes.

Binärform Hier das Programm in für den Computer direkt verwendbarer Form im Gegensatz zum Quellcode. Anders als dort ist hier kaum sichtbar, wie das Programm genau arbeitet, weshalb bei OpenSource-Projekten gerade der Quelltext offen liegt (vgl. OpenSource).

Datenstrom Daten, die beispielsweise während der Ausführung eines Programms fließen, wobei das Ende dieses Flusses nicht absehbar ist.

Debug-Modus Modus, bei dem zusätzliche Informationen angezeigt werden, um dem Programmierer das Auffinden und Beheben von Bugs (kurz *Debugging* oder *Debuggen*), hier insbesondere Programmierfehlern, zu erleichtern. Vgl. Release-Modus.

Debug-Visualisierung Hier eine Visualisierung, die den Benutzer beim Debuggen unterstützt, indem sie relevante Daten zu den übergebenen Bildern anzeigt und diese damit leicht verständlich darstellt.

FAQ Engl. für „Häufig gestellte Fragen“ enthält sie viele Fragen, die besonders neue Benutzer sich stellen, wenn sie anfangen ein Projekt zu nutzen.

Falschfarben Verwendung von Farben ein einem Bild, die sich von den natürlichen, erwarteten Farben stark unterscheiden, um etwa Details hervorzuheben.

Filter In der Bildverarbeitung die Veränderung eines Bildbereiches mithilfe eines bestimmten Algorithmus.

GNU/Linux Das GNU-Betriebssystem in Kombination mit einem Linux-Kern

GNU-Projekt Das Projekt zur Erstellung von GNU-Betriebssystem und Software.

GUI Graphical User Interface, zu deutsch Graphische Benutzeroberfläche. Stellt Funktionen graphisch dar, sodass der Benutzer beispielsweise per Mausklick damit interagieren kann; im Gegensatz zu textbasierten Benutzerschnittstellen (vgl. Interface).

Kompilieren Umwandlung eines Quellcodes in eine für den Computer verständliche Form, mithilfe eines Kompiler genannten Programms.

Matches Durch OpenCV erzeugte Verknüpfungen zwischen zwei Bildbereichen bzw. Bildpixeln, welche vom Benutzer an die API übergeben werden.

Mouse over Information über das Element einer GUI, auf dem der Mauszeiger ruht, wird angezeigt.

Parallelrechner Rechner, in dem mehrere Threads gleichzeitig nebeneinander ausgeführt werden können (vgl. Thread).

OpenCV Test Framework Stellt Funktionen zum Testen zur Verfügung, etwa Überprüfungen, ob zwei Matrizen gleich sind.

Open Source Software, bei welcher der Quellcode frei zugänglich gemacht wird. Dies erlaubt unter anderem die Weiterverwendung und -entwicklung durch andere.

Overhead Zusätzlicher Speicher- oder Zeitaufwand.

proprietär In diesem Zusammenhang Software, die nicht unter einer freien Lizenz steht.

Release-Modus Veröffentlichungs-Modus, in dem keine zusätzlichen Debug-Informationen angezeigt werden, also der Modus, in dem das fertige Programm läuft.

Resource-Leak Das Auftreten der Situation, dass Ressourcen irgendeiner Art (Speicher, Dateien, usw.) zwar alloziert werden, aber nach Verwendung nicht mehr an das System zurückgegeben werden.

Rohdaten Daten, welche direkt und ohne wirkliche Aufarbeitung, aus den vom Entwickler beim API-Aufruf übergebenen Datenstrukturen stammen.

Speicherleck *engl. memory leak* Vgl. Resource-Leak; die Ressource ist in diesem Fall Speicher.

Stand-Alone-Programm Programm, das für sich alleine funktioniert.

Streaming Hier das Weiterlaufen des Datenstroms.

Tab Hier ein registerkartenähnlicher Teil einer GUI.

Thread Ausführungsstrang in einem Programm. Der Begriff wird insbesondere im Zusammenhang mit Mehrfachkernsystemen, wo mehrere Threads gleichzeitig nebeneinander laufen können, oft verwendet.

thread-lokal Auf einen Thread beschränkt (vgl. Thread).

Threadsafety Es ist sichergestellt, das mehrere Threads sich nicht gegenseitig stören, etwa beim Speichern von Daten.

Translation Unit Eine Einheit, die einzeln im Ganzen kompiliert wird (s. Kompilieren); ein Projekt teilt sich meist in mehrere solcher Einheiten auf.

Undefiniertes Verhalten Befehle, deren Verwendung dazu führt, dass der C++-Standard *keinerlei* Verhaltensgarantien irgendeiner Art für das gesamte Programm mehr gibt. Etwas Umgangssprachlich: Der Standard untersagt die Verwendung.

View Zusammengehörige Visualisierungen eines bestimmten OpenCV-Features (oder einer Featureart).

12.2 OpenCV

Weiterführende Informationen sind auf docs.opencv.org zu finden.

adaptiveThreshold OpenCV-Methode, die mittels eines adaptiven Threshold (s. unten) Graustufenbilder in (u.U. invertierte) Binärbilder umwandeln kann.

calcHist Berechnet ein Histogramm.

Canny Kantenerkennung (mithilfe des Canny86-Algorithmus).

Dilatation Berechnung des Bereiches in einem Bild, der von einer Maske abgedeckt wird, wenn sich deren Bezugspunkt (oft der Mittelpunkt) durch den ganzen zu untersuchenden Bildbereich bewegt.

DMatch Klasse für das Matching (vgl. Matches).

Erosion Prüft, ob eine Maske, etwa eine geometrische Figur, vollständig in ein Bild bzw. einen Bildbereich passt und gibt u.U. ein Bild zurück, in dem nur die überdeckten Teile erhalten sind. Bildet zusammen mit der Dilatation zwei der grundlegenden Bildverarbeitungsmethoden.

floodFill Bei dieser Methode wird ein zusammenhängender Bereich des Bildes mit der übergebenen Farbe ausgefüllt.

HoughCircles Findet Kreise in einem Graustufenbild (unter Benutzung der Hough-Transformation).

KeyPoint Klasse, die Daten eines Punktes (Koordinaten, Nachbarschaftsgröße etc.) enthält.

morphologyEx diese Methode von OpenCV erlaubt fortgeschrittene morphologische Transformationen unter Benutzung und mehrfacher Anwendung von Dilatation und Erosion.

ocl Das OCL-Modul stellt Implementierungen von OpenCV-Funktionalität für Geräte, welche die Open Computing Language (kurz OpenCL), ein Interface über Parallelrechner, benutzen, zur Verfügung.

Sobel-Operator Ein Kantenerkennungs-Filter.

Stitching Zusammenfügen mehrerer Bilder mit zusammenhängenden Bereichen an den Rändern zu einem großen Bild, etwa von Einzelfotografien zu einem Panorama.

threshold Diese Methode eröffnet verschiedene Möglichkeiten, die Elemente eines Arrays auf ein bestimmtes Niveau zu trimmen, auf binäre Werte herunterzubrechen und Ähnliches.