פרויקט רכב אוטונומי:

Shoval Nakash 207207838

Adi Gredi 316063353

Liad Arama 206484719

Navigation – Rviz & Ros

תקציר: סקירה כללית של ROS (Robot Operating System) מערכת הפעלה רובוטית בקוד פתוח. ROS היא לא מערכת הפעלה. במובן המסורתי של ניהול ותזמון תהליכים; במקום זאת, הוא מספק שכבת תקשורת מובנית למעלה מערכות ההפעלה המארחות של אשכול מחשוב הטרוגני. ההתייחסות לשאלה כיצד ROS קשור לרובוט קיים במסגרות תוכנה וסקור של חלק מאפשרויות הזמינות לתוכנת יישום המשתמשת ב-ROS.

מערכת הפעלה רובוט (ROS) היא כאמור חבילת תוכנת תווך רובוטית בקוד פתוח . למרות ש-ROS אינה מערכת הפעלה (ROS) אלא קבוצה של מסגרות תוכנה לפיתוח תוכנת רובוט , היא מספקת שירותים המיועדים לאשכול מחשבים הטרוגני כגון הפשטת חומרה , בקרת מכשירים ברמה נמוכה, הטמעת פונקציונליות נפוצה, העברת הודעות בין תהליכים וניהול חבילות. סטים פועלים של תהליכים מבוססי ROS מיוצגים בארכיטקטורת גרפים שבה העיבוד מתרחש בצמתים שעשויים לקבל, לפרסם, נתוני חיישן מרובי , בקרה, מצב, תכנון, מפעיל והודעות אחרות. למרות החשיבות של תגובתיות והשהייה נמוכה בשליטה ברובוט, ROS אינה מערכת הפעלה בזמן אמת (RTOS). עם זאת, ניתן לשלב ROS עם קוד בזמן אמת . היעדר התמיכה במערכות בזמן אמת טופל ביצירת ROS גרסה גדולה של ROS API אשר ינצל את היתרון של ספריות וטכנולוגיות מודרניות עבור ROS ליבה פונקציות והוסף תמיכה בקוד בזמן אמת ובחומרת מערכת משובצת.

האם ROS היא המסגרת הכי טובה?

לא נטען ש-ROS היא המסגרת הטובה ביותר עבור כל תוכנות הרובוטיקה. למעשה, תחום הרובוטיקה רחב מדי לפתרון יחיד ובודד. ROS מתוכנן כדי לענות על סט ספציפי של אתגרים בהם נתקלים בעת הפיתוח בקנה מידה גדול, תנועה זורמת על פני הקישור אלחוטי (באופן איטי), משום שרבים הם נתיבי הודעות כלולים במלואם גם ברשתות המשנה על הסיפון הרובוט או מחוצה לו.

לעומת זאת, peer-to-peer (עמית לעמית) קישוריות, בשילוב עם תוכנת חציצה או "fanout". מודולים במידת הצורך, מונע את הבעיה לחלוטין. הטופולוגיה של peer-to-peer (עמית לעמית) דורשת איזשהו חיפוש מנגנון המאפשר לתהליכים למצוא זה את זה other at runtime (בזמן ריצה) זה מכונה (שירות או מאסטר).

סביבות פיתוח: בעת כתיבת קוד, לאנשים רבים יש העדפות עבור שפות תכנות מסוימות מעל אחרות. העדפות אלו הן תוצאה של פשרות אישיות בין זמן תכנות, קלות ניפוי באגים, תחביר, יעילות other at runtime (בזמן ריצה) ועוד שלל סיבות טכניות ותרבותיות. לסיבות הללו מתוכנן ROS להיות ניטרלי בשפה. הוא תומך בשלב זה בארבע שפות שונות: C++, Python, Octave ו-LISP.
בהמשך יוצרים מחוללי קוד עבור כל שפה נתמכת יישומים מקוריים אשר "מרגישים" כמו אובייקטים מקומיים, וכן עוברים באופן אוטומטי בסדרה וביטול סדרה על ידי ROS as הודעות נשלחות ומתקבלות. קובץ IDL בן שלוש השורות הקודם מתרחב אוטומטית ל-137 שורות של 96 ,++C של Pythonu, 81 שורות של Lisp , ו-99 שורות של Octave. בגלל ההודעות נוצרות אוטומטית מטקסט פשוט. בזמן הכתיבה בסיסי הקוד הידועים מבוססי ROS מכילים למעלה מארבע מאות סוגי הודעות, המעבירות נתונים החל מהזנת חיישנים לזיהוי אובייקטים ועד למפות. התוצאה הסופית היא עיבוד מסרים ניטרלי בשפה תכנית שבה ניתן לערבב ולהתאים שפות שונות.

מונחים: המושגים הבסיסיים של יישום ROS הם צמתים, הודעות, נושאים ושירותים, צמתים הם תהליכים שמבצעים חישוב. ROS הוא תוכנן להיות מודולרי בקנה מידה עדין: מערכת מורכבת בדרך כלל מצמתים רבים. בהקשר זה, המונח "צומת" ניתן להחלפה עם "מודול תוכנה". השימוש במונח "צומת" נובע מהדמיות של מערכות מבוססות ROS.other at runtime (בזמן ריצה): כאשר צמתים רבים פועלים, נוח לעיבוד תקשורת peer-to-peer (עמית לעמית) גרף, עם תהליכים כצמתי גרף peer-to-peer (והעמית לעמית) קישורים כקשתות.

צמתים מתקשרים זה עם זה על ידי העברת הודעות. הודעה היא מבנה נתונים המוקלד בקפדנות. תקן טיפוסים פרימיטיביים (מספר שלם, נקודה צפה, בוליאנית וכו') הם נתמכים, כמו גם מערכים של טיפוסים קבועים ופרימיטיביים. הודעות יכולות להיות מורכבות מהודעות אחרות, וממערכים של הודעות אחרות, מקונן עמוק באופן שרירותי.
צומת שולח הודעה על ידי פרסומו לנושא נתון, שהיא פשוט מחרוזת כגון "אודומטריה" או "מפה". **(א)** צומת שמתעניין בסוג מסוים של נתונים יירשם לנושא המתאים. יכול להיות שיש מספר בו-זמנית מוציאים לאור ומנויים לנושא בודד ויחיד node עשוי לפרסם ו/או להירשם למספר נושאים. **(ב)** באופן כללי, מפרסמים ומנויים אינם מודעים לכל אחד מהם קיומם של אחרים.

התקשורת הפשוטה ביותר היא לאורך צינורות: Microphone, speech recognition, dialog manager, synthesisspeaker, speech. עם זאת, גרפים הם בדרך כלל הרבה יותר מורכבים, לעתים קרובות מכילים מחזורים וחיבורים של אחד לרבים או רבים לרבים. למרות שהמודל מבוסס נושא של פרסום הרשמה להלן: (א) פרדיגמת תקשורת גמישה, ניתוב ה"שידור" שלה התוכנית אינה מתאימה לעסקאות סינכרוניות, מה שיכול לפשט את העיצוב של כמה צמתים. ב-ROS, קוראים לזה שירות, המוגדר על ידי שם מחרוזת וזוג של הודעות מוקלדות בקפדנות: אחת לבקשה ואחת עבור התגובה. זה מקביל לשירותי אינטרנט, שהם מוגדר על ידי URI ויש להם מסמכי בקשה ותגובה

מסוגים מוגדרים היטב. יש לשים לב שבניגוד לנושאים, רק צומת אחד יכול לפרסם שירות בכל שם מסוים: יש רק להיות שירות אחד שנקרא "סווג תמונה". למשל, בדיוק כמו יכול להיות רק שירות אינטרנט אחד בכל URI נתון.

בארבעה מקרים נפוצים נתאר מקרים ותרחישים שנתקלים בהם בעת שימוש במסגרות תוכנה רובוטיות. הארכיטקטורה הפתוחה של ROS מאפשרת את היצירה של מגוון רחב של כלים; בתיאור גישת ה-ROS למקרי שימוש אלה, נציג גם מספר כלים המיועדים לשימוש עם ROS.

(א)איתור באגים בצומת בודד: בעת ביצוע מחקר רובוטיקה, לעתים קרובות ההיקף של החקירה מוגבל לאזור מוגדר היטב של המערכת, כגון צומת שמבצע סוג כלשהו של תכנון, הנמקה, תפיסה או שליטה. עם זאת, כדי לקבל מערכת רובוטית פועלת לניסויים, הרבה מערכת אקולוגית גדולה יותר של תוכנה חייבת להתקיים. למשל, לעשות ניסויי אחיזה מבוססי חזון, דרייברים חייבים לפעול עבור המצלמות והמניפולטורים, וכל מספר של צמתי עיבוד ביניים (למשל מזהה אובייקטים, מיקום גלאים, מתכנני מסלולים) חייבים להיות גם פועלים.

זה מוסיף כמות משמעותית של קושי לאינטגרטיבי מחקר רובוטיקה. ROS נועד למזער את הקושי של איתור באגים בהגדרות כאלה, שכן המבנה המודולרי שלו מאפשר צמתים עוברים פיתוח פעיל כדי לרוץ לצד הקיים, צמתים שפותחו היטב. כי צמתים מתחברים אחד לשני other at runtime (בזמן ריצה), ניתן לשנות את הגרף באופן דינמי.

בתוך הדוגמה קודמת לאחיזה מבוססת חזון, אולי נדרשים תריסר צמתים כדי לספק את התשתית. ניתן להתחיל ולהשאיר את גרף ה"תשתית" הזה other at runtime (בזמן ריצה) במהלך מפגש ניסוי שלם. רק צמתים שעוברים שינוי בקוד המקור צריכים להיות מופעלים מחדש מעת לעת, ובזמן זה ROS מטפל בשקט השינויים בגרף. זה יכול לגרום לעלייה מסיבית בפרודוקטיביות, במיוחד כשהמערכת הרובוטית הופכת מורכבים ומקושרים יותר.

כדי להדגיש, שינוי הגרף ב-ROS פשוט מסתכם בכמויות להתחיל או להפסיק תהליך. בהגדרות איתור באגים, זה מתבצעת בדרך כלל בשורת הפקודה או במפרק באגים. הקלות של הכנסה והסרה של צמתים ממערכת מבוססת ROS פועלת היא אחד החזקים והבסיסיים ביותר של התכונות.

(ב). רישום והשמעה: המחקר בתפיסה רובוטית נעשה לרוב בצורה נוחה ביותר עם נתוני חיישנים מתועדים, כדי לאפשר שליטה השוואות של לוגריתמים שונים ולפשט את ההליך הניסיוני. ROS תומך בגישה זו על ידי מתן פונקציונליות רישום והשמעה גנרית. כל ROS ניתן לזרוק את זרם ההודעות לדיסק ולהפעיל אותו מאוחר יותר.

חשוב לציין, כל זה יכול להיעשות בשורת הפקודה; אין צורך לשנות את קוד המקור של חלקים כלשהם של תוכנה בגרף. לדוגמה, גרף הרשת הבא יכול להיות הגדרה מהירה לאיסוף מערך נתונים עבור ניתוח מרחק חזותי מחקר: camera, logger, visualizer, robot.

ניתן להפעיל בחזרה את dump ההודעות שהתקבל גרף אחר, המכיל את הצומת בפיתוח:

Logger - vision research – visualizer

כמו קודם, ניתן לבצע מופע צמתים פשוט על ידי השקת תהליך; ניתן לעשות זאת בשורת הפקודה, מאתר באגים, מתסריט וכו'. כדי להקל על רישום וניטור של מערכות מבוזרות על פני מארחים רבים, ספריית rosconsole מתבסס על מערכת log 4cxx של פרויקט Apache כדי לספק גישה נוחה וממשק רישום אלגנטי, המאפשר לנתב הודעות אבחון בסגנון printf דרך הרשת ליחידה אחת זרם שנקרא rosout.

(ג) תת מערכות ארוזות: בתחומים מסוימים של מחקר רובוטיקה, כמו רובוט פנימי ניווט, הבשילו עד לנקודה שבה "מתוך אלגוריתמים של box" יכולים לעבוד בצורה סבירה. ROS ממנפת האלגוריתמים המיושמים בפרויקט Player לספק מערכת ניווט, המייצרת את הגרף הזה: למרות שניתן להפעיל כל צומת משורת הפקודה, הקלדה חוזרת מערכת ניווט, ROS מספקת כלי שנקרא roslaunch,

שקורא תיאור XML של גרף ומציג הגרף על האשכול, אופציונלי על מארחים ספציפיים. חווית משתמש קצה בהשקת מערכת הניווט אז מסתכם roslaunch navstack.xml ו-Ctrl-C יחיד יסגור בחן את כל חמשת התהליכים. פונקציונליות זו יכולה גם לסייע באופן משמעותי בשיתוף ובשימוש חוזר של הדגמות גדולות של מחקר רובוטיקה אינטגרטיבי, כמו ההגדרה והפירוק של מערכות מבוזרות גדולות יכולות להיות משוכפל בקלות.

(**ד**). **פיתוח שיתופי** בשל ההיקף העצום של רובוטיקה ובינה מלאכותית, שיתוף פעולה בין חוקרים הכרחי על מנת לבנות מערכות גדולות. כדי לתמוך בפיתוח שיתופי, מערכת התוכנה ROS מאורגנת בחבילות. ההגדרה של "חבילה" היא פתוחה בכוונה: ROS החבילה למעשה היא פשוט ספרייה המכילה קובץ XML תיאור החבילה וציון כל תלות. אוסף של חבילות ROS הוא עץ ספריות עם ROS חבילות על העלים: מאגר חבילות ROS עשוי כך מכילים סכמה מורכבת באופן שרירותי של ספריות משנה. לדוגמה, למאגר ROS אחד יש ספריות שורש כולל "vision", "nav" ו-"motion planning", שכל אחת מהן מכילה חבילות רבות בתור ספריות משנה. ROS מספק כלי עזר בשם rospack לשאילתות ובדוק את עץ הקוד, חפש תלות, מצא חבילות לפי השם וכו'.

קבוצה של הרחבות מעטפת הנקראות rosbash היא מסופק מטעמי נוחות, האצת ניווט שורת הפקודה של המערכת. כלי השירות rospack נועד לתמוך בפיתוח בו-זמני על פני מספר מאגרי חבילות ROS. משתני סביבה משמשים להגדרת השורשים של מקומי. עותקים של מאגרי חבילות ROS, וסריקות rospack עצי החבילה לפי הצורך. בנייה רקורסיבית, נתמכת על ידי כלי השירות rosmake, לאפשר ספרייה חוצה חבילות תלות.

האופי הפתוח של חבילות ROS מאפשר נהדר שונות במבנה ובמטרה שלהם: כמה חבילות ROS לעטוף תוכנות קיימות, כגון Player או Open CV, אוטומציה של ה-builds שלהן וייצוא הפונקציונליות שלהן. כמה חבילות בונות צמתים לשימוש בגרפי ROS, חבילות אחרות לספק ספריות וקובצי הפעלה עצמאיים

ואחרים לספק סקריפטים כדי להפוך הדגמות ובדיקות לאוטומטיות. המערכת אריזה נועדה לחלק את המבנה של תוכנה מבוססת ROS לנתחים קטנים הניתנים לניהול, כך, שכל אחד מהם ניתן לתחזק ולפתח לפי לו"ז משלו וע"י צוות מפתחים משלו.

ויזואליזציה וניטור: בזמן תכנון וניפוי באגים של תוכנות רובוטיקה, זה לעתים קרובות נחוץ כדי לצפות במצב כלשהו בזמן המערכת רץ. למרות ש-printf היא טכניקה מוכרת עבור איתור באגים בתוכניות במכונה אחת, טכניקה זו יכולה קשה להרחיב למערכות מבוזרות בקנה מידה גדול, וכן יכול להיות מסורבל לניטור למטרות כלליות. במקום זאת, ROS יכול לנצל את האופי הדינמי של גרף קישוריות כדי "להקיש" על כל זרם הודעות המערכת. יתר על כן, הניתוק בין מפרסמים ומנויים מאפשרים יצירת תכלית כללית חזותיים. ניתן לכתוב תכניות פשוטות אשר נרשמות לשם נושא מסוים ולתכנן סוג מסוים של נתונים, כגון סריקות לייזר או תמונות. עם זאת, חזק יותר קונספט היא תכנית הדמיה המשתמשת בתוסף ארכיטקטורה: זה נעשה בתוכנית rviz, כלומר מופץ עם ROS. ניתן ליצור לוחות ויזואליזציה באופן דינמי כדי להציג מגוון גדול של סוגי נתונים, כגון כמו תמונות, point clouds (ענני נקודה), פרימיטיביים גיאומטריים geometric primitives (כגון אובייקט תוצאות זיהוי such as object recognition results) עיבוד תנוחות רובוט ומסלולים וכו'.

ניתן לכתוב PLUGINS בקלות כדי להציג יותר סוגי נתונים. יציאת ROS מקורית מסופקת עבור Python, שפה בעלת טיפוס דינמי התומכת בבדיקה פנימה. באמצעות, כלי עזר רב עוצמה בשם rostopic נכתב כדי לסנן הודעות באמצעות ביטויים המסופקים בשורת הפקודה, וכתוצאה מכך "הקשה על הודעה" הניתנת להתאמה אישית מיידית אשר יכול להמיר כל חלק מכל זרם נתונים לזרם טקסט. ניתן להעביר זרמי טקסט אלו לכלי שורת פקודה אחרים של UNIX כגון grep, sed ו-awk, כדי ליצור מורכבות כלי ניטור מבלי לכתוב שום קוד. באופן דומה, כלי שנקרא rxplot מספק את הפונקציונליות של אוסילוסקופ וירטואלי, משרטט

כל משתנה בזמן אמת כסדרת זמן, שוב באמצעות התבוננות פנימית של Python והערכת ביטוי.

הרכב הפונקציונליות : ב-ROS, "ערימה" של תוכנה היא מקבץ צמתים העושה משהו שימושי, כפי שהודגם בניווט דוגמא. כפי שתואר קודם, ROS מסוגל ליצור מופע אשכול צמתים עם פקודה בודדת, פעם האשכול מתואר בקובץ XML.
עם זאת, לפעמים מרובים רצויות מופעים של אשכול. לדוגמה, בניסויים רב רובוטים, יהיה צורך בערימת ניווט עבור כל רובוט במערכת, ורובוטים עם פלג גוף עליון דמוי אדם סביר להניח שיהיה צורך ליצור שני בקרי זרוע זהים. ROS תומך בכך על ידי התרת צמתים ו-roslaunch שלם קבצי תיאור אשכול שיש לדחוף לתוך מרחב שמות צאצא, ובכך להבטיח שלא יהיו התנגשויות שמות.
נושאים ושמות שירותים, ללא צורך בשינוי כלשהו לקוד של הצומת או האשכול.
הגרף הבא מראה מערכת בקרה רב-רובוטית היררכית שנבנתה פשוט על ידי יצירת ערימות ניווט מרובות, כל אחת מהן מרחב השמות שלהם: הגרף הקודם נוצר באופן אוטומטי על ידי כלי rxgraph, שיכול לבדוק ולנתר כל ROS גרף (בזמן ריצה). הפלט שלו הופך צמתים כאליפסות, נושאים כמו ריבועים, וקישוריות כקשתות.

NAVIGATION

Nav הוא היורש הרוחני הנתמך באופן מקצועי של מחסנית הניווט ROS. פרויקט זה מבקש למצוא דרך בטוחה לגרום לרובוט נייד לעבור להשלמת משימות מורכבות באמצעות סוגים רבים של סביבות ומחלקות של של רובוטים. לא רק שהוא יכול לעבור מנקודה A לנקודה B, אלא שהוא יכול לקבל תנוחות מתווך, ולייצג סוגים אחרים של משימות כמו מעקב אחר אובייקט ועוד. Nav היא מסגרת ניווט ברמה ייצור ואיכותית שאמינה על ידי 50+ חברות ברחבי העולם.

הוא מספק תפיסה, תכנון, בקרה, לוקליזציה, ויזואליזציה ועוד הרבה יותר כדי לבנות מערכות אוטונומיות אמינות במיוחד. זה ישלים מודלים סביבתיים מנתוני חיישנים, תכנון נתיב דינמי, חישוב מהירויות עבור מנועים, ימנע מכשולים, ייצוג אזורים ואובייקטים סמנטיים ויבנה התנהגויות רובוטים ברמה גבוהה יותר. למידע נוסף על פרויקט זה, כגון פרויקטים קשורים, שימוש ברובוטים, השוואת ROS1 ותחזוקה, ראה אודות ויצירת קשר. למידע נוסף על מושגי ניווט ו-ROS, ראה מושגי ניווט.

Nav משתמש בעצי התנהגות כדי ליצור התנהגות ניווט מותאמת וחכמה באמצעות תזמור שרתים מודולריים רבים ובלתי תלויים. ניתן להשתמש בשרת משימות כדי לחשב נתיב, מאמץ לשלוט, התאוששות או כל משימת ניווט אחרת הקשורה. שרתים נפרדים אלה מתקשרים עם עץ ההתנהגות (BT) דרך ממשק ROS כגון שרת פעולה או שירות. רובוט עשוי להשתמש בעצי התנהגות רבים ושונים כדי לאפשר לרובוט לבצע סוגים רבים של משימות ייחודיות.

התרשים שלהלן ייתן לך מבט ראשוני טוב על המבנה של Nav. הערה: אפשר לקבל תוספים מרובים לבקרים, מתכננים ושחזורים בכל אחד מהשרתים שלהם עם תוספים BT תואמים. זה יכול לשמש ליצירת התנהגויות ניווט קונטקסטואליות. אם תרצה לראות השוואה בין פרויקט זה לבין ניווט ROS

התשומות הצפויות ל-Nav הן טרנספורמציות TF התואמות ל-REP-105, מקור מפה אם נעשה שימוש בשכבת העלות הסטטית, קובץ BT XML, וכל מקור נתוני חיישן רלוונטי. לאחר מכן הוא יספק פקודות מהירות תקפות עבור המנועים של רובוט הולונומי או לא הולונומי. כרגע אנו תומכים בכל סוגי הרובוטים העיקריים: סוגי בסיס הולונומי, הנעה דיפרנציאלית, רגליים וסוגי אקרמן (כמו מכונית)! אנו תומכים בהם באופן ייחודי עם רובוטים עגולים ושרירותיים לבדיקת התנגשות SE2.

סיכום: אפשר לומר שתכנון ROS נועד לתמוך בתפיסת פיתוח תוכנה מודולרי המבוסס על כלים. כפי שפורטו להלן. ויש צפייה שניתן יהיה להרחיב ולבנות לעצב ולפתח כלים נוספים ע"י גורמים מהתחום כאלה ואחרים במטרה לבנות מערכות תוכנת רובוט שיכולות להיות שימושיות למגוון פלטפורמות כמו חומרה, מחקר ועוד.

היתרונות העיקריים של שימוש ב-ROS: המפתח חייב להיות אחראי רק על פיתוח הקוד המקושר לאפליקציה מכיוון ש-ROS מציע רמה גבוהה מאוד של הפשטה ביחס לחומרה. בנוסף, מכיוון שמפותחות הפונקציות הנפוצות ביותר, ניתן להשתמש בהן ואין צורך להמציא את הגלגל מחדש. צמתים שונים הפועלים על פלטפורמות חומרה שונות אך במערכת ROS אחת, מציעה גמישות רבה. את הצומת שפותח עבור אפליקציה ספציפית ניתן להתאים בקלות לאפליקציה אחרת, ואפילו לעשות שימוש חוזר ישירות ללא צורך בשינוי דבר בקוד. יש להתקין רק את חבילות התוכנה הנדרשות עבור האפליקציה. ואין צורך להתקין את כל חבילות ה-ROS.

ביבולגרפיה :

http://wiki.ros.org/navigation

/https://navigation.ros.org

https://www.theconstructsim.com/robotigniteacademy_learnros/ros

/-courses-library/ros-courses-ros-navigation-in-5-days

http://wiki.ros.org/rviz

http://wiki.ros.org/rviz/UserGuide