

ORB SLAM2

מדריך התקנה

נכתב ע"י: גל ברק

התאמות לגרסה 20.4: ליעם ונונו

מבוסס על המדריך שנכתב
במעבדת **RBD** באוניברסיטת
חיפה

גרסה 3.1

02/10/20

תוכן עניינים

3.....	פרק 1 - הקדמה
4.....	פרק 2 - הכנות
5.....	שלב 1 - התקנת מכונה וירטואלית
6.....	שלב 2 - התקנת מערכת ההפעלה
12.....	פרק 3 - התקנת הספריות
13.....	שלב 1 - התקנת ספריית פיתוח
13.....	שלב 2 - התקנת GIT
13.....	שלב 3 - התקנת Python
14.....	שלב 4 - התקנת OpenCV
15.....	שלב 5 - התקנת ספריית Pangolin
15.....	שלב 6 - התקנת Eigen3
16.....	שלב 7 - התקנת OpenVSLAM & BLAS & LAPACK
17.....	פרק 4 - ביצוע Build
19.....	פתרון תקלות

רשימת עדכונים

מס'	גרסה	עדכונים
1	1.0	כתיבת המסמך
2	2.0	הוספת הוראות התקנה ל- OpenCV ו- Ros Melodic
3	2.1	תיקון התקנת pangolin
4	2.2	הוספת הצעת פתרון לשגיאת Internal compiler error בזמן ביצוע build ל- ORB_SLAM2
5	3.0	התאמת ההתקנה ל- ubuntu 20.04 - ותיקון שגיאות הקורות בעקבות גרסת ubuntu (השינויים בגרסה זו נכתבו ע"י ליעם ונונו)
6	3.1	תיקונים נוספים והתאמות ל- ubuntu 20.04

פרק 1 - הקדמה

ORB-SLAM2 is a real-time SLAM library for **Monocular**, **Stereo** and **RGB-D** cameras that computes the camera trajectory and a sparse 3D reconstruction (in the stereo and RGB-D case with true scale). It is able to detect loops and relocalize the camera in real time. We provide examples to run the SLAM system in the [KITTI dataset](#) as stereo or monocular, and in the [TUM dataset](#) as RGB-D or monocular. We also provide a ROS node to process live monocular or RGB-D streams. **The library can be compiled without ROS.** ORB-SLAM2 provides a GUI to change between a *SLAM Mode* and *Localization Mode*, see section 9 of this document.

Notice for ORB-SLAM Monocular users: The monocular capabilities of ORB-SLAM2 compared to [ORB-SLAM Monocular](#) are similar. However in ORB-SLAM2 we apply a full bundle adjustment after a loop closure, the extraction of ORB is slightly different (trying to improve the dispersion on the image) and the tracking is also slightly faster. The GUI of ORB-SLAM2 also provides you new capabilities as the *modes* mentioned above and a reset button. We recommend you to try this new software :)

ORB-SLAM2 is released under a [GPLv3 license](#). For a list of all code/library dependencies (and associated licenses), please see [Dependencies.md](#).

For a closed-source version of ORB-SLAM2 for commercial purposes, please contact the authors: orbslam (at) unizar (dot) es.

If you use ORB-SLAM2 in an academic work, please cite:

```
@article{murTRO2015,
  title={ {ORB-SLAM}: a Versatile and Accurate Monocular {SLAM} System},
  author={Mur-Artal, Ra'ul, Montiel, J. M. M. and Tard'os, Juan D.},
  journal={IEEE Transactions on Robotics},
  volume={31},
  number={5},
  pages={1147--1163},
  doi = {10.1109/TRO.2015.2463671},
  year={2015}
}
```

Original installation guide website https://github.com/faresfaresCS/ORB_SLAM2

פרק 2 – הכנות

פרק זה עוסק בהתקנת התוכנות טרם התקנת הספריות עבור ORB_SLAM2.

בפרק זה נעבור על התקנת המכונה הווירטואלית עליה נריץ את מערכת ההפעלה Ubuntu

לאחר מכן נגדיר את מערכת ההפעלה על המכונה הווירטואלית ונכין את התשתית להתקנת הספריות.

הערה: פרק זה כולל הורדה של תוכנות והתקנה שלהן על המחשב שלכם. מומלץ להוריד את התוכנות העדכניות ביותר מהאתרים הרשמיים. בכל שלב כתבתי באיזו גרסה השתמשתי. כמו כן וודאו כי המחשב שלכם עומד בדרישות המינימום להתקנת התוכנות.

שלב 1 – התקנת מכונה וירטואלית

במידה ומותקנת אצלכם מכונה וירטואלית ומערכת הפעלה Ubuntu, ניתן לעבור [לשלב 3](#). במידה ומותקנת אצלכם מכונה וירטואלית מוכנה להתקנת Ubuntu יש לעבור [לשלב 2](#).

ראשית נתקין את המכונה הווירטואלית Oracle VM VirtualBox 6.1.12

הורדת ההתקנה תבוצע מהאתר <https://www.virtualbox.org/wiki/Downloads>

שימו לב כי אתם מורידים בהתאם למערכת ההפעלה הראשית שלכם (host).

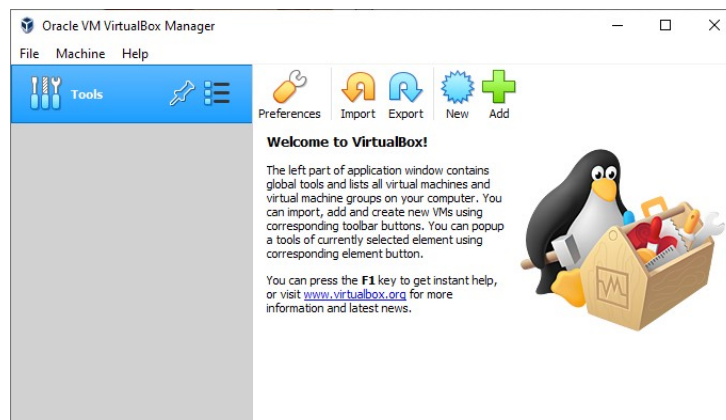
במדריך זה אעשה שימוש בגרסה 6.1.12.

לאחר ההורדה נפתח את ההתקנה ונבצע התקנה לפי ההוראות באשף ההתקנה.

לבסוף נסמן את האפשרות Start Oracle VM



לאחר לחיצה על Finish ייפתח החלון הראשי של המכונה הווירטואלית.



כתב: גל ברק

שלב 2 – התקנת מערכת ההפעלה

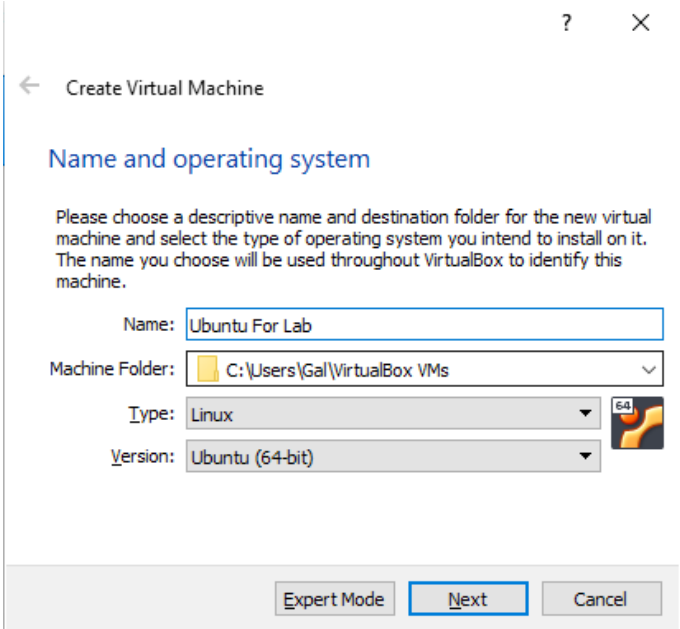
את מערכת ההפעלה Ubuntu נוריד מהאתר <https://ubuntu.com/download/desktop>

את שלב זה כתבתי ע"פ ההתקנה של גרסה 20.04 (לכן גם בהתאם התמונות והתפריטים)

יש להוריד את גרסת ה- desktop בהתאם למערכת ההפעלה שלכם (32 / 64 bit).

לאחר ההורדה נפתח את קובץ ההתקנה בעזרת כונן וירטואלי (מומלץ להשתמש ב- Virtual CloneDrive)

כעת נפתח את המכונה הווירטואלית (VirtualBox) ונלחץ על כפתור New ולאחר מכן במסך שייפתח נמלא את הפרטים הרלוונטיים




← Create Virtual Machine

Name and operating system

Please choose a descriptive name and destination folder for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.

Name:

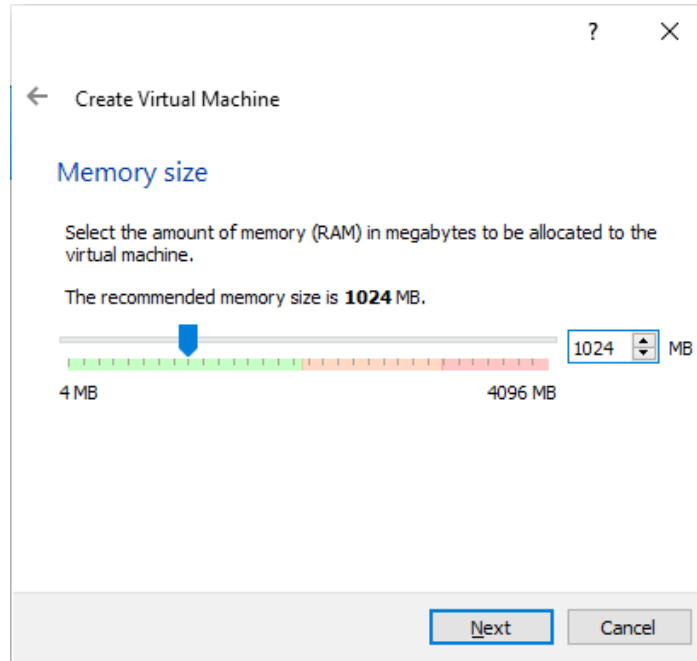
Machine Folder:

Type: 

Version:

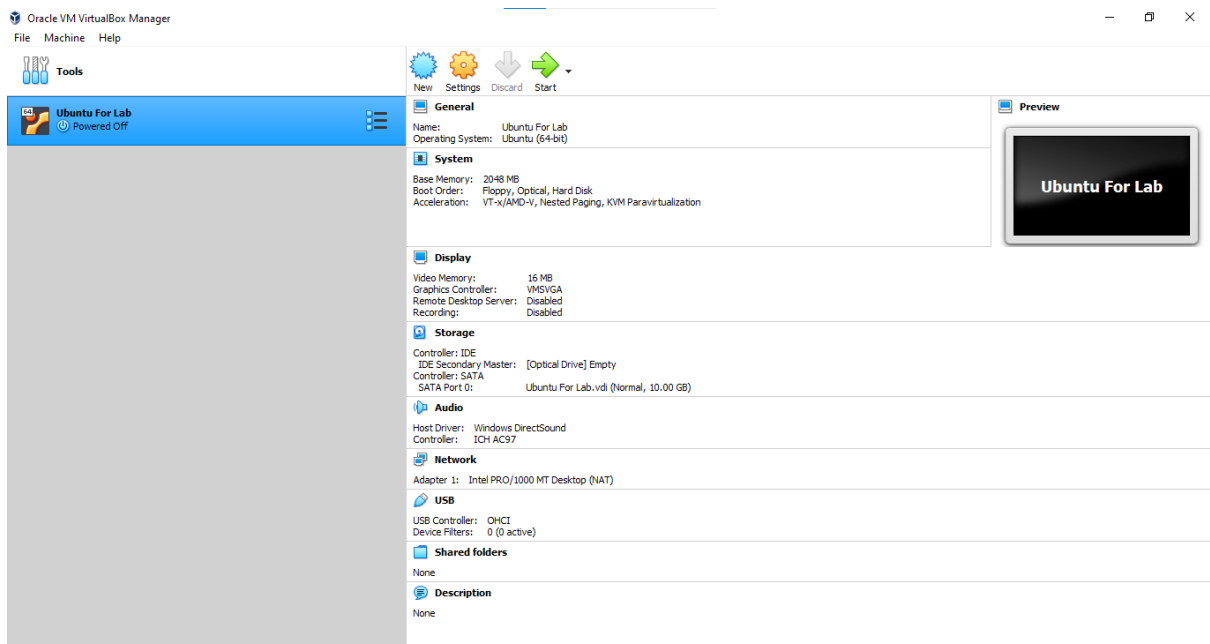
שימו לב כי בשדה Version צריך למלא את הגרסה אותה הורדתם.

לאחר לחיצה על next נגדיר כמה זיכרון יוקצה עבור מערכת ההפעלה הזו
שימו לב כי מערכת ההפעלה הזו רצה במקביל למערכת הראשית שלכם ולכן לא כדאי לחרוג
 מההגדרות המומלצות.

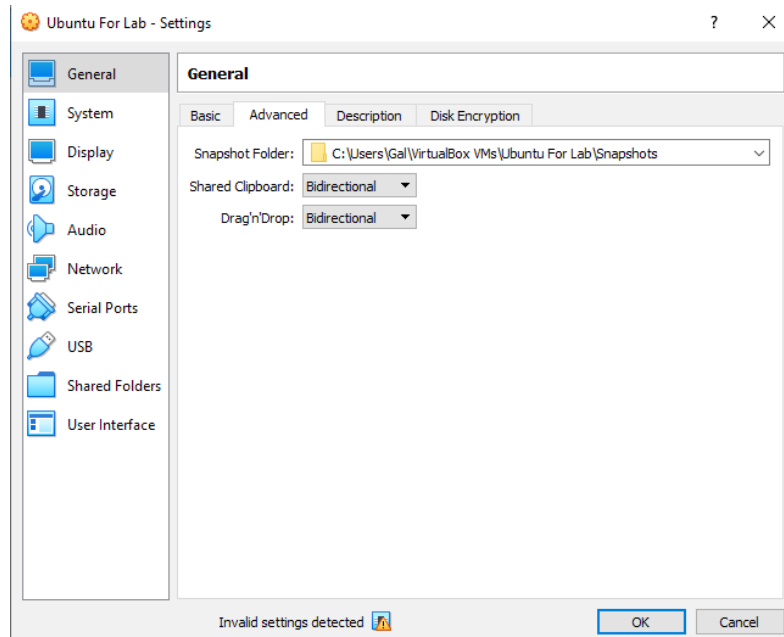


בשלב הבא נגדיר את הזיכרון הקשיח שיוקצה למכונה הווירטואלית (באופן דינאמי או מוגדר מראש).
 המלצה שלי להגדיר הקצאה דינאמית של לפחות 30GB.

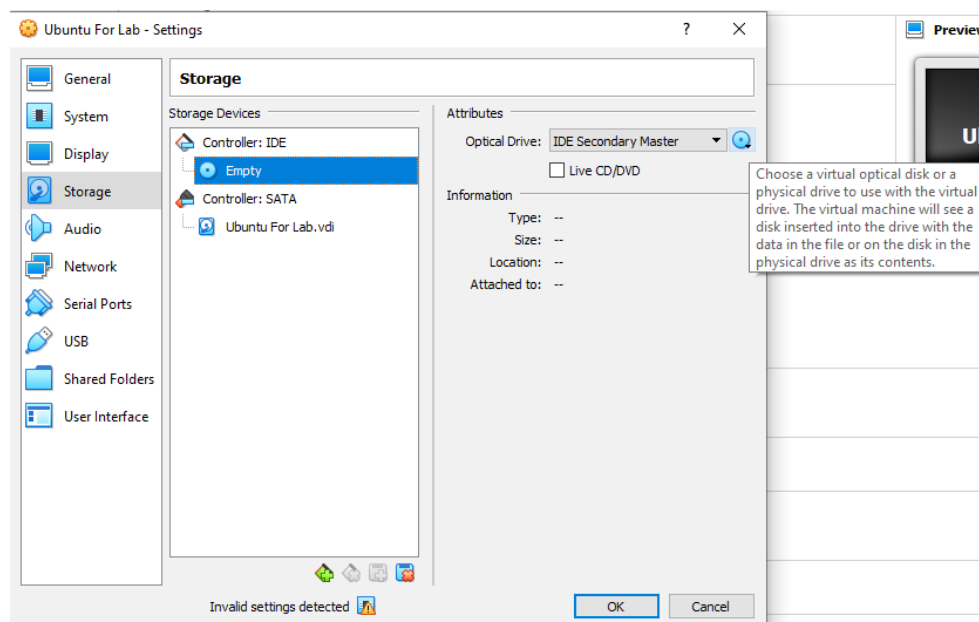
כעת נוכל לראות כי נוספה לנו מערכת הפעלה חדשה



לפני שנריץ את מערכת ההפעלה נוכל להגדיר תחת תפריט Settings מספר הגדרות כגון שיתוף קבצים בין מערכת ההפעלה למכונה הווירטואלית.



מומלץ להגדיר כ- Bidirectional כדי לאפשר העברת קבצים נוחה אל / מ המכונה הווירטואלית. ניתן גם לשחק עם הגדרות נוספות אבל נתמקד כרגע בשדה Storage



נסמן את השדה Empty ולאחר מכן נלחץ על הדיסק הכחול מצד ימין

במסך שנפתח נבחר את קובץ ה- iso שהורדנו

Name	Date modified	Type
ubuntu-20.04-desktop-amd64	20/07/2020 21:01	Virtual CloneDrive

כתב: גל ברק

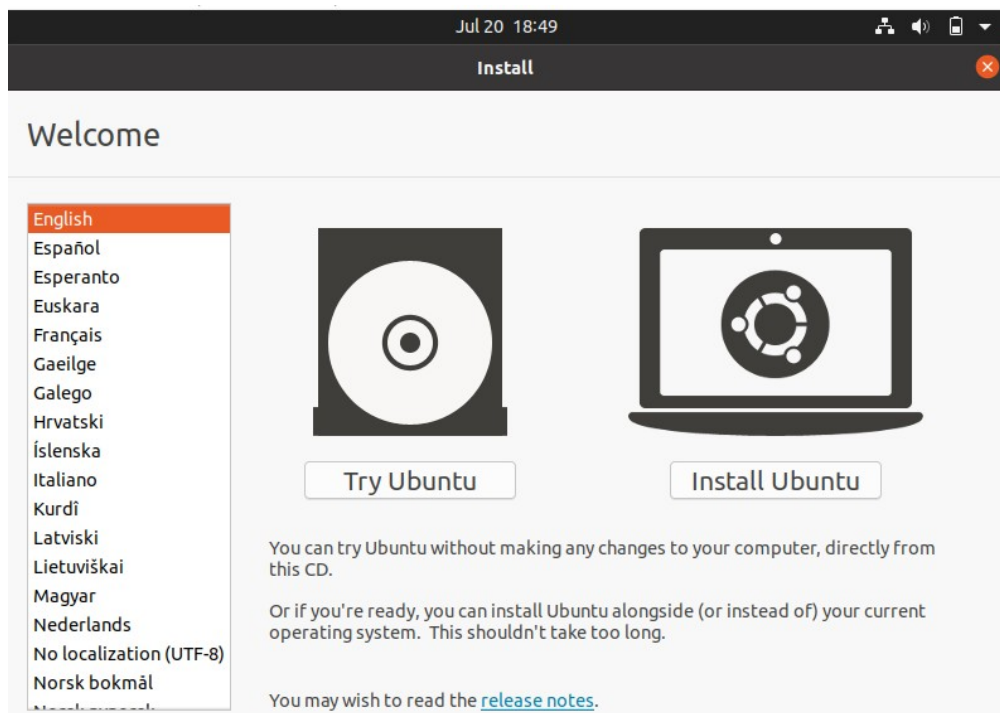
לאחר מכן נראה שהשדה Empty התחלף לשם הקובץ שלנו ונלחץ על OK.

כעת מערכת ההפעלה שלנו מוכנה להרצה ראשונה.

נריץ את מערכת ההפעלה. בהרצה הראשונה המערכת תטען הגדרות

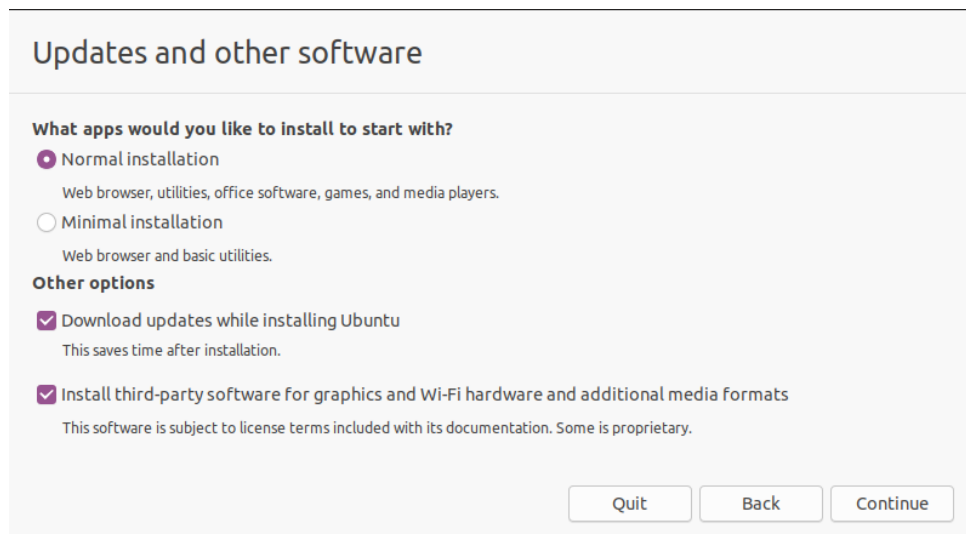


לאחר מכן נצטרך לבצע התקנה הראשונה.



נגדיר שפת התקנה ונלחץ על Install. לאחר מכן נמשיך עם אשף ההתקנה.

אם אין לכם שימוש מתקדם או ידע קודם, מומלץ להשאיר את כל ההגדרות כפי שהן למעט הוספת סימון על התקנת חבילות צד שלישי



Updates and other software

What apps would you like to install to start with?

- ☒ Normal installation
Web browser, utilities, office software, games, and media players.
- ☐ Minimal installation
Web browser and basic utilities.

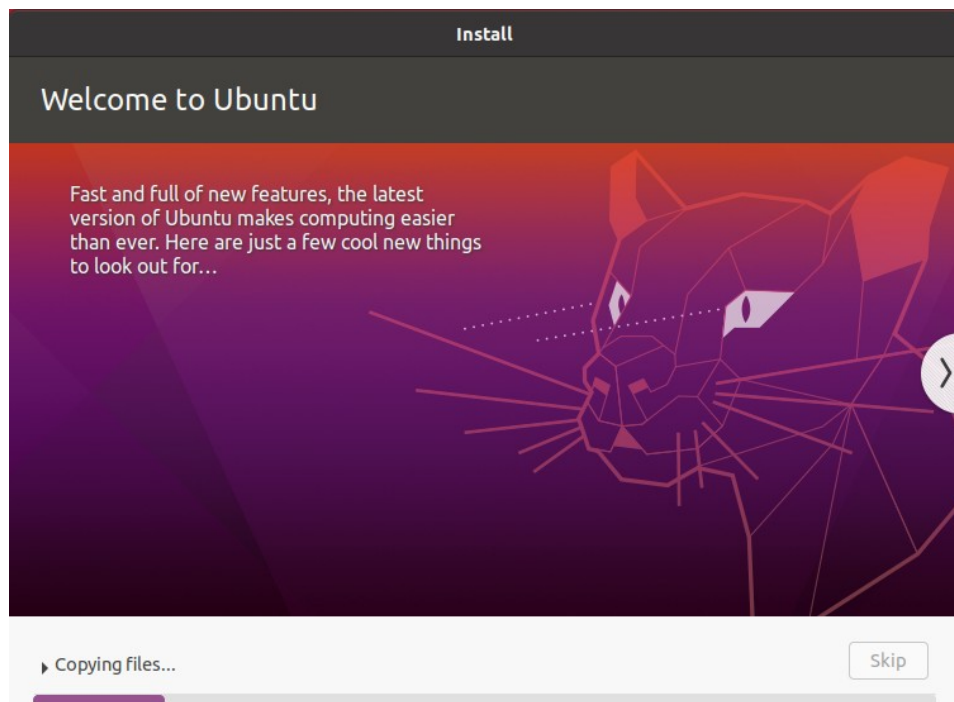
Other options

- ☒ Download updates while installing Ubuntu
This saves time after installation.
- ☒ Install third-party software for graphics and Wi-Fi hardware and additional media formats
This software is subject to license terms included with its documentation. Some is proprietary.

Quit Back Continue

לבסוף נלחץ על Install now ונאשר.

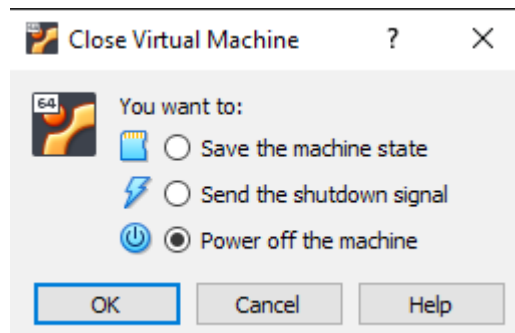
עכשיו המערכת תאפשר לנו להגדיר משתמש בהרשאות (root) .admin. נמלא את הפרטים לפי רצוננו ונמשיך לביצוע ההתקנה.



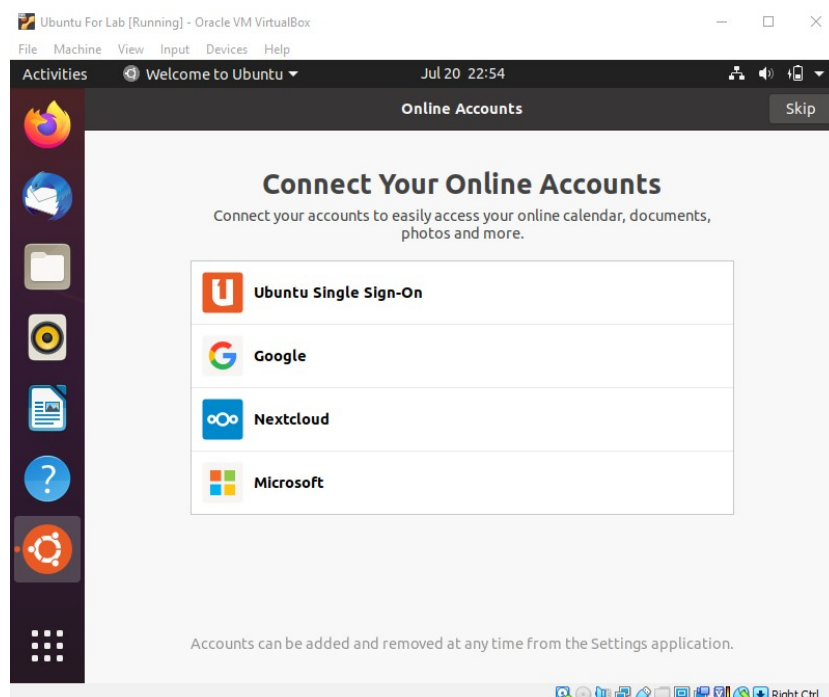
בסוף ההתקנה נלחץ על Restart now וכאשר נקבל את ההודעה הזו



נסגור את החלון ונסמן את האפשרות



כעת במסך הראשי של ה-VirtualBox נריץ שוב את מערכת ההפעלה שהתקנו ע"י לחיצה על Start. בהרצה זו מערכת ההפעלה תכניס אותנו למסך ההתחברות. נכניס את פרטי ה-admin שהגדרנו קודם.



כתב: גל ברק

פרק 3 – התקנת הספריות

בשעה טובה סיימנו להתקין את הסביבה הווירטואלית וכעת אפשר להתחיל להתקין את הספריות עבור ORB_SLAM2.

שימו לב שההתקנה ארוכה ודורשת זמן. במהלך ההתקנה תדרשו להוריד חבילות, לבצע התקנות ובעיקר להמתין בסבלנות עד שכל התהליכים יסתיימו.

שימו לב לגרסאות שאתם מתקינים. אם הגרסה שונה מזו שבמדריך, זה עשוי לסבך אתכם בהמשך.

את כל ההתקנות נבצע בחלון ה-Terminal של Ubuntu

בהצלחה!

שלב 1 – התקנת ספריית פיתוח

במהלך התקנת הספריות נדרש לקמפל ספריות קוד. את תהליך הקומפילציה נבצע באמצעות קומפיילר GCC.

נפתח Terminal ונרשום את הפקודה הבאה

```
sudo apt install build-essential
```

נזין סיסמה של משתמש root ונמשיך בהתקנה

בסיום נוודא כי גרסת הקומפיילר שלנו היא לפחות 9.3.0 בעזרת הפקודה

```
gcc -v
```

שלב 2 – התקנת Git

כדי שנוכל לייבא ספריות קוד מ- git בעזרת פקודות clone נדרש להריץ ב- Terminal את הפקודה

```
sudo apt install git
```

שלב 3 – התקנת Python

חלק מגרסאות Ubuntu מגיעות עם גרסת python מובנת. ראשית נבדוק את הגרסה המותקנת אצלנו.

```
python3 --version
```

או

```
python3 -v
```

אם התוצאה גבוהה מ- 3.7.x ניתן לדלג [לשלב הבא](#) (המדריך נבדק עם גרסה 3.6.9)

כדי להתקין / לעדכן גרסת python נריץ את הפקודות הבאות

```
sudo apt update
sudo apt install software-properties-common
sudo add-apt-repository ppa:deadsnakes/ppa
sudo apt update
```

כעת נריץ בדיקה ונוודא כי מותקנת לנו גרסה עדכנית של python 3

```
python3 --version
```

או

```
python3 -v
```

שלב 4 – התקנת OpenCV

התקנת OpenCV תבוצע לפי המדריך הבא (אנא קראו את ההערות לפני ביצוע המדריך):

[/https://www.learnopencv.com/install-opencv3-on-ubuntu](https://www.learnopencv.com/install-opencv3-on-ubuntu)

הערות למדריך:

- במהלך המדריך תיתקלו באופציות להתקנה גם בעזרת python3 וגם בעזרת python2. תבצעו את כל המדריך רק על python3 (python3, pip3 וכו...)
- את המדריך יש לבצע עד שלב 5.3
- לפני ביצוע שלב 5 במדריך נכנס לקובץ -
opencv/modules/videoio/src/cap_ffmpeg_impl.hpp
בקובץ זה נוסיף בתחילת הקוד את שלושת השורות הבאות:

```
#define AV_CODEC_FLAG_GLOBAL_HEADER (1 << 22)
#define CODEC_FLAG_GLOBAL_HEADER AV_CODEC_FLAG_GLOBAL_HEADER
#define AVFMT_RAWPICTURE 0x0020
```

הערה חשובה: שימו לב לרווחים שבין המילים בשורות (בעיקר לרווח שבין AV_CODEC_FLAG_GLOBAL_HEADER לבין (1 >> 22)). אצל רוב האנשים הרווח הקיים נעלם לאחר ההעתקה וגורם לתכנית ליפול).

לאחר מכן נכנס לקובץ – cv2.cpp/opencv/modules/python/src2 – ונשנה את סוג המשתנה בשורה מ-char ל-const char. לאחר שינויים אלו נבצע את שלב 5

- חלק מספריות הבסיס במדריך כבר הותקנו במסגרת מדריך זה. הספריות לא יותקנו פעמיים, לכן תוכלו פשוט להריץ את כל הפקודות או לדלג את אלו שכבר התקנתם.
- המדריך כולל התקנות אופציונאליות שלא חייבים להתקין. שימו לב להנחיות.

בסוף ההתקנה לפי המדריך (לאחר שלב 5.3) נריץ ב-Terminal את הפקודה הבאה ונוודא שגרסת OpenCV היא לפחות 2.4.3 (המדריך נבדק על גרסה 3.3.1)

```
python3 -c "import cv2; print(cv2.__version__)"
```

שלב 5 – התקנת ספריית Pangolin

Pangolin זו ספרייה המאפשרת את ממשק המשתמש. כדי להתקין אותה נייבא אותה מ- git

```
git clone https://github.com/stevenlovegrove/Pangolin.git
```

כעת נתחיל לתקין את הספריות

```
sudo apt install libgl1-mesa-dev
sudo apt install libglew-dev
sudo apt install cmake
```

לבסוף נעשה build לספרייה

```
cd Pangolin
mkdir build
cd build
cmake ..
cmake --build
make
sudo make install
```

לאחר ביצוע make יש לוודא כי libpangolin.so נמצא בתיקייה

```
/Pangolin/build/src
```

במידה ולא, יש להעתיק אותו מתיקייה

```
/Pangolin/src
```

שלב 6 – התקנת Eigen3

Eigen זו ספריית קוד עבור חישובים אלגבריים.

הערה: במידה והתקנתם את ההרחבות בהתקנת OpenCV אחת מהן כללה התקנה של eigen3 ניתן לבדוק זאת בעזרת הפקודה

```
pkg-config --modversion eigen3
```

במידה ולא, התקינו את הספרייה עם הפקודה הבאה:

```
sudo apt install libeigen3-dev
```

בתחילת ההתקנה נדרש להכניס סיסמה למשתמש ולאשר התקנה.

לאחר מכן נבדוק את גרסת הספרייה המותקנת (לפחות 3.1.0) המדריך נבדק עם גרסה 3.3.4

שלב 7 – התקנת OpenVSLAM & BLAS & LAPACK

```
sudo apt-get install libblas-dev  
sudo apt-get install liblapack-dev  
sudo apt-get install libboost-all-dev
```

פרק 4 – ביצוע Build

סיימנו להתקין ולהגדיר את כל הספריות והתוכנות הנדרשות. כעת נוריד את הפרויקט מה- Git ונבצע Build.

שימו לב כי בתהליך ה- Build ייתכנו שגיאות שנובעות מהתקנות לא תקינות בשלב 3. במידה ויהיו תקלות, עקבו אחריהן ובצעו את התיקונים הנדרשים.

ראשית נבצע Clone ל- repository ב- git

```
git clone https://github.com/faresfaresCS/ORB_SLAM2.git
```

לאחר מכן נכנס לקובץ – ORB_SLAM2/include/LoopClosing.h ובשורה 50 נשנה את הכתוב מ:

```
typedef map<KeyFrame*, g2o::Sim3, std::less<KeyFrame*>,
Eigen::aligned_allocator<std::pair<const KeyFrame*, g2o::Sim3> > >
KeyFrameAndPose;
```

ל:

```
typedef map<KeyFrame*, g2o::Sim3, std::less<KeyFrame*>,
Eigen::aligned_allocator<std::pair<KeyFrame* const, g2o::Sim3> > >
KeyFrameAndPose;
```

כעת נריץ את הפקודה

```
cd ORB_SLAM2
chmod +x build.sh
./build.sh
```

לאחר מכן יתחיל תהליך של build לפרויקט

```
Configuring and building Thirdparty/DBoW2 ...
-- The C compiler identification is GNU 9.3.0
-- The CXX compiler identification is GNU 9.3.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found OpenCV: /usr (found version "4.2.0")
-- Configuring done
-- Generating done
-- Build files have been written to: /home/gal/Desktop/ORB_SLAM2/Thirdparty/DBoW2/build
Scanning dependencies of target DBoW2
[ 14%] Building CXX object CMakeFiles/DBoW2.dir/DBoW2/BowVector.cpp.o
[ 42%] Building CXX object CMakeFiles/DBoW2.dir/DBoW2/ScoringObject.cpp.o
[ 57%] Building CXX object CMakeFiles/DBoW2.dir/DBoW2/FORB.cpp.o
[ 71%] Building CXX object CMakeFiles/DBoW2.dir/DUtils/Random.cpp.o
[ 71%] Building CXX object CMakeFiles/DBoW2.dir/DBoW2/FeatureVector.cpp.o
[ 85%] Building CXX object CMakeFiles/DBoW2.dir/DUtils/Timestamp.cpp.o
```

בסיום התהליך (עשוי לקחת זמן) ואם הכל הלך תקין, תקבלו את ההודעה הבאה

```
Configuring and building ORB_SLAM2 ...
mkdir: cannot create directory 'build': File exists
Build type: Release
-- Using flag -std=c++11.
-- Boost version: 1.65.1
-- Found the following Boost libraries:
--   serialization
-- Configuring done
-- Generating done
-- Build files have been written to: /home/gal/ORB_SLAM2/build
[ 66%] Built target ORB_SLAM2
[ 73%] Built target rgbd_tum
[ 80%] Built target mono_kitti
[ 86%] Built target stereo_kitti
[ 93%] Built target mono_tum
[100%] Built target bin_vocabulary
Converting vocabulary to binary
BoW load/save benchmark
Loading from text: 42.63s
Saving as binary: 0.86s
gal@VirtualBox:~/ORB_SLAM2$
```

כלומר התהליך הסתיים ואתם מוכנים להתחיל לעבוד.

מזל טוב! סיימתם להתקין את ORB SLAM2!

פתרון תקלות

1. Internal compiler error while compiling תקלת

במידה ומתקבלת התקלה הבאה

```
c++: internal compiler error: Killed (program cc1plus)
Please submit a full bug report,
with preprocessed source if appropriate.
See <file:///usr/share/doc/gcc-7/README.Bugs> for instructions.
CMakeFiles/ORB_SLAM2.dir/build.make:86: recipe for target 'CMakeFiles/ORB_SLAM2
.dir/src/Tracking.cc.o' failed
make[2]: *** [CMakeFiles/ORB_SLAM2.dir/src/Tracking.cc.o] Error 4
make[2]: *** Waiting for unfinished jobs....
```

נפתח את הקובץ build.sh בתיקייה ORB_SLAM2 ונחליף את השורה `make -j$(nproc)` ל-`make`
נשמור את הקובץ ונבצע קריאה נוספת ל-

```
./build.sh
```