一、VISP 安装步骤及配置

1. Win7 版本为专业版 service packet 1。

2. VISP 版本为 3.2.1（2019.11.3）。

3. 将 VISP 装在 D 盘，按 WIN+R 快捷键，输入 cmd，打开命令提示符，输入 D：进入 D 盘，输入 setx VISP_WS "D:\visp-ws"后按 Enter 键创建 VISP_WS 工作环境变量，输入 mkdir D:\visp-ws 后按 Enter 键生成相应的文件夹。

4. VS 版本为 VS2017 专业版，安装 Development Desktop C++，注意右边的 Windows 8.1 SDK 和 UCRT SDK 项要选中安装，否则后面 CMake 会出错。

5. 直接双击 cmake-3.14.3-win64-x64 安装 CMake。

6. 直接双击 Git-2.21.0-64-bit，全部选中默认选项进行安装。

7. 双击 opencv-4.1.1-vc14_vc15，解压到 D:\visp-ws\opencv，并将文件夹重命名为 opencv-4.1.1，在命令提示符中输入 setx OpenCV_DIR "D:\visp-ws\opencv-4.1.1\build"，将 D:\visp-ws\opencv-4.1.1\build\x64\vc15\bin 添加进系统路径变量中。

8. 将 eigen-3.3.7 文件夹复制到 D:\visp-ws 路径下，在命令提示符中依次输入 mkdir D:\visp-ws\eigen-3.3.7\build-vc15，cd D:\visp-ws\eigen-3.3.7\build-vc15，cmake -G "Visual Studio 15 2017 Win64" %VISP_WS%\eigen-3.3.7 -DCMAKE_INSTALL_PREFIX=%VISP_WS%\eigen-3.3.7\build-vc15\install，cmake --build . --config Release --target install，setx Eigen3_DIR "%VISP_WS%\eigen-3.3.7\build-vc15\install\share\eigen3\cmake"。

9. 双击 Intel.RealSense.SDK-WIN7 默认选项安装，该安装包包含了驱动和 SDK，不过仅对 D400 系列的相机有效。Win10 系统下要依次安装驱动和 SDK 文件。安装完成后要将 dll 的路径 C:\Program Files (x86)\Intel RealSense SDK 2.0\bin\x64 添加到系统路径变量中。

10. 双击 Basler_pylon_5.0.12.11830 安装 Basler 相机驱动，注意版本用的是 5.0.12 版本。安装过程中注意要选"Developer"，安装路径选择默认路径 C:\Program Files\Basler\pylon 5\。

11. 将 visp 文件夹复制到 D:\visp-ws 路径下，打开 CMake (cmake-gui)，source code 路径填 D:\visp-ws\visp，编译路径填 D:\visp-ws\visp-build-vc15，点击 Configure，选择 Visual Studio 15 2017 作为生成器，

选择 x64 作为生成平台，点击 Finish 开始进行配置，配置完后点击 Generate，等待生成完成后即可关闭 CMake。

12. 双击 D:\visp-ws/visp-build-vc15 路径下的 VISP.sln 文件，选择 Release 配置，点击菜单中的“生成>生成解决方案”，等待生成后，点击 INSTALL 项目，右键点击生成。同样地，如何选择 Debug 配置，后两步操作相同。

13. 将 D:\visp-ws\visp-build-vc15\install\x64\vc15\bin 添加到系统路径变量中。

14. 在命令提示符中输入 setx VISP_DIR "D:\visp-ws\visp-build-vc15\install"设置 VISP_DIR 环境变量。

15. Win10 的安装步骤跟 Win7 的相同，区别仅在于在 Win10 中安装 VS2017 时不必安装 Windows 8.1 SDK 和 UCRT SDK。

16. Windows 8.1 SDK 和 UCRT SDK 项，另外相机的驱动和 SDK 版本可能不一样。

17. 注意事项：
源代码及项目的配置不能轻易改，如符号格式等。cmake 文件夹中的 FindRealsense2 有作更改，安装目录改成了 C:\Program Files (x86)\Intel RealSense SDK 2.0 (Win7)。

二、软件运动控制器及驱动器调试

1. 零点时各轴编码器位置（单位 Inc）：
   1 轴：103319
   2 轴：92992
   3 轴：116630
   4 轴：31953
   5 轴：111221
   6 轴：91157

2. 各轴减速比：
   1 轴：80008：1000（80.008）
   2 轴：99902：1000（99.902）
   3 轴：78433：1000（78.433）
   4 轴：50001：1000（50.001）
   5 轴：64001：1000（64.001）
   6 轴：40000：1000（40.000）

3. 耦合比：

6 轴=6 轴+0.01248916*5 轴

4. 电机 1 角度对应编码器 Inc 数：
   131072/360=364.08888889
   编码器每 1Inc 对应的角度：
   360/131072=0.00274658

5. 上位机发送 1 脉冲给驱动器，编码器位置增加 1Inc。

6. IPMCStopAxis 单轴停止。
   IPMCStopAllAxis 所有轴停止。
   IPMCChangeTargetPosition 在线实时改变位置（目前不可用）。
   IPMCEmergeStop 紧急停止所有轴。
   IPMCSetVelCommand 速度模式下更改速度。
   单轴运动时设置目标速度用 SetAxisVel。
   读取逻辑轴位置 GetAxiPos、速度 GetAxiVel。
   读取驱动器位置 GetDriverPos、速度 GetDriverVel。

7. 插补时设置目标速度 ContiSetTargetVel 和 SetInterpolationVel 均可；

   停止连续插补时，立即停止 1 和减速停止 0 无区别，都表现为减速停止；

   ContiSetTargetVel 和 ContiOpenList 顺序可以调换；

   前瞻段数？如何确定误差参数？坐标系号？

   如果不对插补速度设置时，GetAxisVel 的值默认为 8；单轴运动时 GetAxisVel 的值默认为 19，均表现为很慢；

   ContiSetTargetVel 的速度设为 10000 时，逻辑轴速度 GetAxisVel 的值为 4082，而驱动器的实际速度反馈值 GetDriverVel 和驱动器的速度指令值为 4082000 左右；

   逻辑计算位置 GetAxiPos 和 GetDriverPos 驱动器的实际位置反馈值相近，说明控制器是将逻辑计算位置发送给驱动器的；

   单轴运动时，SetAxisVel 的速度设为 10000，逻辑轴速度 GetAxisVel 的值为 10000，而驱动器的实际速度反馈值 GetDriverVel 和驱动器的速度指令值为 10000000 左右；

   说明在位置控制模式下，驱动器的实际速度反馈值 GetDriveVel 和驱动器的速度指令值均为控制器逻辑轴速度 GetAxisVel 的 1000 倍；

   停止连续插补后要停止所有轴 IPMCStopAllAxis 才可执行单轴运动。

8. 在速度控制模式下：
   速度设为（2 的编码器位数次方值时，单位为 Inc/s），电机转速为 1s/r；

   驱动器上的速度目标值跟上位机 IPMCSetVelCommand 指令发送的值相同，而速度指令值是根据加速度和目标速度等计算得到的值（未运动时在零上下波动）；

   当速度设为零时，位置却在递增？当速度设为-10000 左右时，位置才大致不变；

   停止所有轴指令 IPMCStopAllAxis 对速度模式不可用；

   驱动器允许的最大速度为 Vmax=524288000/8388608*2*PI(rad/s)；

   应添加 GetAxisCommandMode()的 API，确保每个轴都进入 CSV 模式。

三、视觉伺服项目建立

（注意：以下说明文档是为 Debug 模式调试时用，如果要转为 Release 模式，要将所有的 lib 和 dll 文件换成 Release 版本。另外，要确保所有目录的路径是否正确。）

1. 新建 C++空项目，将 IPMCMOTION.h 和 vpRobotKawasaki.h 头文件以及 servoKawasakiPBVS.cpp 和 vpRobotKawasaki.cpp 源文件放在工程目录下，并在工程中添加。

2. 打开项目属性窗口，下拉左边 C/C++属性，点击常规，在右边的附加包含目录中添加：
   E:\VISP\servoKawasaki\servoKawasakiPBVS\servoKawasakiPBVS、
   E:\VISP\install\include、D:\visp-ws\opencv-4.1.1\build\include、
   C:\Program Files (x86)\Intel RealSense SDK 2.0 (Win7)\include；

   下拉左边链接器属性，点击常规，在右边附加库目录中添加：
   E:\VISP\servoKawasaki\servoKawasakiPBVS\servoKawasakiPBVS、
   E:\VISP\install\x64\vc15\lib；

   点击左边输入，在右边的附加依赖项中添加：
   IPMCMOTION.lib
   visp_tt_mi321d.lib
   visp_tt321d.lib
   visp_mbt321d.lib
   visp_klt321d.lib
   visp_imgproc321d.lib

visp_ar321d.lib
visp_robot321d.lib
visp_gui321d.lib
visp_vs321d.lib
visp_detection321d.lib
visp_sensor321d.lib
C:\Program Files (x86)\Intel RealSense SDK 2.0 (Win7)\lib\x64\realsense2.lib
C:\Program Files (x86)\Microsoft SDKs\Windows\v7.1A\Lib\x64\Gdi32.Lib
visp_vision321d.lib
visp_visual_features321d.lib
visp_me321d.lib
visp_blob321d.lib
visp_io321d.lib
visp_core321d.lib
D:\visp-ws\opencv-4.1.1\build\x64\vc15\lib\opencv_world411d.lib
winmm.lib
ws2_32.lib

3. 在 IPMCMOTION.h 头文件中添加#include <Windows.h>以防报错。

4. vpRobotKawasaki.h 是在 vpRobotTemplate.h 的基础上修改的，vpRobotTemplate 全改为 vpRobotKawasaki，增加了 PI、Deg2Rad、Rad2Deg、ROBOT_DOF 宏定义，connect()、getMotorVelocity()、getAxisVelocity()、isSingular()函数，以及脉冲当量、连杆参数、零点时实际转过的角度、各轴最大最小限位、零点时各轴编码器读数、各轴减速比、各轴运动方向、编码器分辨率等变量

5. vpRobotKawasaki.cpp 是在 vpRobotTemplate.cpp 的基础上修改的，vpRobotTemplate 全改为 vpRobotKawasaki，增加了#include <vpRobotKawasaki.h>和#include <IPMCMOTION.h>，添加语句 using namespace std;，具体定义了 init、vpRobotKawasaki()、~vpRobotKawasaki()、set_eMc()、get_eMc()、connect()、get_eJe()、setCartVelocity()、setJointVelocity()、setVelocity()、getJointPosition()、getPosition()、setRobotState()、getAxisVelocity()、getMotorVelocity()、isSingular()等函数；其他函数 get_fJe()、getDisplacement()、setPosition()为非必需，暂未定义；

   值得注意的是，可以在 init 中修改最大速度，connect()函数要判断驱动器是否全部使能且转为 CSV 控制模式；get_eJe()函数采用矢量积法求雅克比矩阵 fJe 后，还应左乘一个转换矩阵，公式可见 VISP，最后才得到 eJe（此变量反映末端执行器坐标相对其当前自身坐标作微小移动时，各关节应该转多少角度）；读取或设置电机位置或速度的函数均需将脉冲数通过相

关公式转换为对应的角度或转速；isSingular()函数用于判断雅克比矩阵 eJe 是否接近奇异值，但尚未验证其正确性。

6. servoKawasakiPBVS.cpp 是在 servoFrankaPBVS.cpp 的基础上修改的，vpRobotFranka 全改为 vpRobotKawasaki，增加了#include <IPMCMOTION.h>和#include < RobotKawasaki.h>，删除 franka 相关语句，vpopt_eMc_filename = "eMc.yaml"、opt_plot = true、opt_tagSize = 0.096;、添加判断 robot.connect() == EXIT_FAILURE，相机帧率改为 60，vpCameraParameters cam(611.1634091225, 612.4700916733, 345.5597302213, 235.2964336455, 0.0743932293, -0.0725463672);修改 plotter 和 displayFrame，增加 vpColVector qdot_Axis = robot.getAxisVelocity(vpRobot::CAMERA_FRAME, v_c);vpColVector qdot_Motor = robot.getMotorVelocity(vpRobot::CAMERA_FRAME, v_c);可注释掉 display_point_trajectory(I, vip, traj_vip);，增加 catch (const std::exception &e)。可以通过修改 vpAdaptiveGain lambda(3, 0.4, 30); 或 task.setLambda(0.8);的参数控制速度的大小。

7. servoKawasakiIBVS 项目的建立与 servoKawasakiPBVS 相同，其中修改的内容也一致。

四、相机畸变和手眼关系标定

1. 在开始视觉伺服任务之前，需要在 Win10 电脑上，配合 KEBA 控制器，让机器人末端执行器上的相机运动到多个不同位置采集标定板图像。

2. 相机畸变标定依次运行：tutorial-grabber-realsense --seqname chessboard-%02d.png --record 1、calibrate-camera default-chessboard.cfg 后得到相机内参描述文件 camera.xml。

3. 手眼关系标定依次运行：tutorial-franka-acquire-calib-data --ip 10.0.0.2、tutorial-chessboard-pose --square_size 0.027 --input image-%d.png、tutorial-hand-eye-calibration --ndata <number of images or robot poses>后得到表示手眼关系的配置文件 eMc.yaml。

4. 将 camera.xml 和 eMC.yaml 文件放在 servoKawasakiPBVS 或 servoKawasakiIBVS 工程目录下，若直接运行 exe 文件则需将其放在 Debug 或 Release 目录下，确保能够正确加载。

五、KEBA 控制器的配合

1. 该部分在 Win10 电脑中实现，过程较为复杂，暂时不做展开介绍。