

# Proprioceptive Visual Tracking of a Humanoid Robot Head Motion

João Peixoto<sup>1</sup>, Vitor Santos<sup>1,2</sup>, and Filipe Silva<sup>1,2</sup>

<sup>1</sup> Universidade de Aveiro,

<sup>2</sup> Institute for Electronics Engineering and Informatics of Aveiro - IEETA  
{joao.peixoto,vitor,fmsilva}@ua.pt

**Abstract.** This paper addresses the problem of measuring a humanoid robot head motion by fusing inertial and visual data. In this work, a model of a humanoid robot head, including a camera and inertial sensors, is moved on the tip of an industrial robot which is used as ground truth for angular position and velocity. Visual features are extracted from the camera images and used to calculate angular displacement and velocity of the camera, which is fused with angular velocities from a gyroscope and fed into a Kalman Filter. The results are quite interesting for two different scenarios and with very distinct illumination conditions. Additionally, errors are introduced artificially into the data to emulate situations of noisy sensors, and the system still performs very well.

**Keywords:** Kalman filter, SURF, Inertial sensor, Humanoid balance

## 1 Introduction

Humanoid robot balance is a relevant and complex problem despite the continuous progresses done by the research community in the field. The electromechanical limitations and flaws of the structures and actuators make the problem even harder by forbidding the existence of reliable models. Hence, to control these robots, a rich set of sensors, both proprioceptive and exteroceptive, are required. For balance measurement, and ultimately its control, force and/or inertial sensors are usually combined, but the trend may be pushed further, and the combination of more sensors altogether promises more robust and effective representations of the robot internal state, and its state on the environment.

In that line, this paper presents a method for combining inertial and visual data to measure, and later control, humanoid robot motion, namely at the level of the head, where cameras are normally placed, along possibly with inertial sensors. Combining such data may be presented as a challenging task, however it may be very valuable in several contexts, including motion learning. This sensorial merging can increase the robustness of the information since various sensors will feed data into a single model. This work was developed for PHUA (Project Humanoid at the University of Aveiro) with the intention of aiding the progress made to date to this project[1].

The method presented in this paper is based on the Kalman Filter tool, which will combine both sets of data (visual and inertial) into a single representation that describes the movement of the robot, namely its head. The line of focus of this research is to monitor especially the angular position of the robots head relatively to the gravity vector.

## 2 Related Work

The creation of visual-inertial systems has been a complex field of study for some time. Many researchers use this approach in order to improve data that inertial systems can't achieve alone. Commonly, visual-inertial systems are used to enhance odometry [2] and in aiding navigation [3]. This kind of data merging has been also applied to humanoid platforms in ego-motion estimation [4]. Often, this approaches use Extended Kalman Filter (EKF) needing complex formulas in order to describe the systems and the relation between sets of data. Commonly, these research activities are based on experiments, lacking a robust ground truth in order to compare the results to. This paper aims to simplify the problem in order to understand how simplistic a model can be and yet create a functional system with improved data, relying on a ground truth, in order to objectively compare the results with and without the merging of different types of data.

## 3 Experimental Setup

For this paper, we used a trustworthy tool in order to obtain a reliable ground truth, as well as repeatable experiments. This can be accomplished with an industrial manipulator. The manipulator used is a FANUC 200iB, which presents a high repeatability ( $\pm 0.10mm$ ), and has six degrees of freedom. For this fact we are able to perform and reproduce testing trials with high repeatability rates as well as acquiring extremely reliable data from its end-effector. In this case we aimed to obtain the orientation of the FANUC end-effector. The software was developed in language C++ in the environment ROS [5] (Robotic Operating System) and makes use of ROS Topics and Bags.

## 4 Proposed Approach

### 4.1 Kalman Filter

The Kalman Filter is a powerful statistic tool [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29] [30] [31] [32] [33] [34] [35] [36] [37] [38] [39] [40] [41] [42] [43] [44] [45] [46] [47] [48] [49] [50] [51] [52] [53] [54] [55] [56] [57] [58] [59] [60] [61] [62] [63] [64] [65] [66] [67] [68] [69] [70] [71] [72] [73] [74] [75] [76] [77] [78] [79] [80] [81] [82] [83] [84] [85] [86] [87] [88] [89] [90] [91] [92] [93] [94] [95] [96] [97] [98] [99] [100] [101] [102] [103] [104] [105] [106] [107] [108] [109] [110] [111] [112] [113] [114] [115] [116] [117] [118] [119] [120] [121] [122] [123] [124] [125] [126] [127] [128] [129] [130] [131] [132] [133] [134] [135] [136] [137] [138] [139] [140] [141] [142] [143] [144] [145] [146] [147] [148] [149] [150] [151] [152] [153] [154] [155] [156] [157] [158] [159] [160] [161] [162] [163] [164] [165] [166] [167] [168] [169] [170] [171] [172] [173] [174] [175] [176] [177] [178] [179] [180] [181] [182] [183] [184] [185] [186] [187] [188] [189] [190] [191] [192] [193] [194] [195] [196] [197] [198] [199] [200] [201] [202] [203] [204] [205] [206] [207] [208] [209] [210] [211] [212] [213] [214] [215] [216] [217] [218] [219] [220] [221] [222] [223] [224] [225] [226] [227] [228] [229] [230] [231] [232] [233] [234] [235] [236] [237] [238] [239] [240] [241] [242] [243] [244] [245] [246] [247] [248] [249] [250] [251] [252] [253] [254] [255] [256] [257] [258] [259] [260] [261] [262] [263] [264] [265] [266] [267] [268] [269] [270] [271] [272] [273] [274] [275] [276] [277] [278] [279] [280] [281] [282] [283] [284] [285] [286] [287] [288] [289] [290] [291] [292] [293] [294] [295] [296] [297] [298] [299] [300] [301] [302] [303] [304] [305] [306] [307] [308] [309] [310] [311] [312] [313] [314] [315] [316] [317] [318] [319] [320] [321] [322] [323] [324] [325] [326] [327] [328] [329] [330] [331] [332] [333] [334] [335] [336] [337] [338] [339] [340] [341] [342] [343] [344] [345] [346] [347] [348] [349] [350] [351] [352] [353] [354] [355] [356] [357] [358] [359] [360] [361] [362] [363] [364] [365] [366] [367] [368] [369] [370] [371] [372] [373] [374] [375] [376] [377] [378] [379] [380] [381] [382] [383] [384] [385] [386] [387] [388] [389] [390] [391] [392] [393] [394] [395] [396] [397] [398] [399] [400] [401] [402] [403] [404] [405] [406] [407] [408] [409] [410] [411] [412] [413] [414] [415] [416] [417] [418] [419] [420] [421] [422] [423] [424] [425] [426] [427] [428] [429] [430] [431] [432] [433] [434] [435] [436] [437] [438] [439] [440] [441] [442] [443] [444] [445] [446] [447] [448] [449] [450] [451] [452] [453] [454] [455] [456] [457] [458] [459] [460] [461] [462] [463] [464] [465] [466] [467] [468] [469] [470] [471] [472] [473] [474] [475] [476] [477] [478] [479] [480] [481] [482] [483] [484] [485] [486] [487] [488] [489] [490] [491] [492] [493] [494] [495] [496] [497] [498] [499] [500] [501] [502] [503] [504] [505] [506] [507] [508] [509] [510] [511] [512] [513] [514] [515] [516] [517] [518] [519] [520] [521] [522] [523] [524] [525] [526] [527] [528] [529] [530] [531] [532] [533] [534] [535] [536] [537] [538] [539] [540] [541] [542] [543] [544] [545] [546] [547] [548] [549] [550] [551] [552] [553] [554] [555] [556] [557] [558] [559] [560] [561] [562] [563] [564] [565] [566] [567] [568] [569] [570] [571] [572] [573] [574] [575] [576] [577] [578] [579] [580] [581] [582] [583] [584] [585] [586] [587] [588] [589] [590] [591] [592] [593] [594] [595] [596] [597] [598] [599] [600] [601] [602] [603] [604] [605] [606] [607] [608] [609] [610] [611] [612] [613] [614] [615] [616] [617] [618] [619] [620] [621] [622] [623] [624] [625] [626] [627] [628] [629] [630] [631] [632] [633] [634] [635] [636] [637] [638] [639] [640] [641] [642] [643] [644] [645] [646] [647] [648] [649] [650] [651] [652] [653] [654] [655] [656] [657] [658] [659] [660] [661] [662] [663] [664] [665] [666] [667] [668] [669] [670] [671] [672] [673] [674] [675] [676] [677] [678] [679] [680] [681] [682] [683] [684] [685] [686] [687] [688] [689] [690] [691] [692] [693] [694] [695] [696] [697] [698] [699] [700] [701] [702] [703] [704] [705] [706] [707] [708] [709] [710] [711] [712] [713] [714] [715] [716] [717] [718] [719] [720] [721] [722] [723] [724] [725] [726] [727] [728] [729] [730] [731] [732] [733] [734] [735] [736] [737] [738] [739] [740] [741] [742] [743] [744] [745] [746] [747] [748] [749] [750] [751] [752] [753] [754] [755] [756] [757] [758] [759] [760] [761] [762] [763] [764] [765] [766] [767] [768] [769] [770] [771] [772] [773] [774] [775] [776] [777] [778] [779] [780] [781] [782] [783] [784] [785] [786] [787] [788] [789] [790] [791] [792] [793] [794] [795] [796] [797] [798] [799] [800] [801] [802] [803] [804] [805] [806] [807] [808] [809] [810] [811] [812] [813] [814] [815] [816] [817] [818] [819] [820] [821] [822] [823] [824] [825] [826] [827] [828] [829] [830] [831] [832] [833] [834] [835] [836] [837] [838] [839] [840] [841] [842] [843] [844] [845] [846] [847] [848] [849] [850] [851] [852] [853] [854] [855] [856] [857] [858] [859] [860] [861] [862] [863] [864] [865] [866] [867] [868] [869] [870] [871] [872] [873] [874] [875] [876] [877] [878] [879] [880] [881] [882] [883] [884] [885] [886] [887] [888] [889] [890] [891] [892] [893] [894] [895] [896] [897] [898] [899] [900] [901] [902] [903] [904] [905] [906] [907] [908] [909] [910] [911] [912] [913] [914] [915] [916] [917] [918] [919] [920] [921] [922] [923] [924] [925] [926] [927] [928] [929] [930] [931] [932] [933] [934] [935] [936] [937] [938] [939] [940] [941] [942] [943] [944] [945] [946] [947] [948] [949] [950] [951] [952] [953] [954] [955] [956] [957] [958] [959] [960] [961] [962] [963] [964] [965] [966] [967] [968] [969] [970] [971] [972] [973] [974] [975] [976] [977] [978] [979] [980] [981] [982] [983] [984] [985] [986] [987] [988] [989] [990] [991] [992] [993] [994] [995] [996] [997] [998] [999] [1000] [1001] [1002] [1003] [1004] [1005] [1006] [1007] [1008] [1009] [1010] [1011] [1012] [1013] [1014] [1015] [1016] [1017] [1018] [1019] [1020] [1021] [1022] [1023] [1024] [1025] [1026] [1027] [1028] [1029] [1030] [1031] [1032] [1033] [1034] [1035] [1036] [1037] [1038] [1039] [1040] [1041] [1042] [1043] [1044] [1045] [1046] [1047] [1048] [1049] [1050] [1051] [1052] [1053] [1054] [1055] [1056] [1057] [1058] [1059] [1060] [1061] [1062] [1063] [1064] [1065] [1066] [1067] [1068] [1069] [1070] [1071] [1072] [1073] [1074] [1075] [1076] [1077] [1078] [1079] [1080] [1081] [1082] [1083] [1084] [1085] [1086] [1087] [1088] [1089] [1090] [1091] [1092] [1093] [1094] [1095] [1096] [1097] [1098] [1099] [1100] [1101] [1102] [1103] [1104] [1105] [1106] [1107] [1108] [1109] [1110] [1111] [1112] [1113] [1114] [1115] [1116] [1117] [1118] [1119] [1120] [1121] [1122] [1123] [1124] [1125] [1126] [1127] [1128] [1129] [1130] [1131] [1132] [1133] [1134] [1135] [1136] [1137] [1138] [1139] [1140] [1141] [1142] [1143] [1144] [1145] [1146] [1147] [1148] [1149] [1150] [1151] [1152] [1153] [1154] [1155] [1156] [1157] [1158] [1159] [1160] [1161] [1162] [1163] [1164] [1165] [1166] [1167] [1168] [1169] [1170] [1171] [1172] [1173] [1174] [1175] [1176] [1177] [1178] [1179] [1180] [1181] [1182] [1183] [1184] [1185] [1186] [1187] [1188] [1189] [1190] [1191] [1192] [1193] [1194] [1195] [1196] [1197] [1198] [1199] [1200] [1201] [1202] [1203] [1204] [1205] [1206] [1207] [1208] [1209] [1210] [1211] [1212] [1213] [1214] [1215] [1216] [1217] [1218] [1219] [1220] [1221] [1222] [1223] [1224] [1225] [1226] [1227] [1228] [1229] [1230] [1231] [1232] [1233] [1234] [1235] [1236] [1237] [1238] [1239] [1240] [1241] [1242] [1243] [1244] [1245] [1246] [1247] [1248] [1249] [1250] [1251] [1252] [1253] [1254] [1255] [1256] [1257] [1258] [1259] [1260] [1261] [1262] [1263] [1264] [1265] [1266] [1267] [1268] [1269] [1270] [1271] [1272] [1273] [1274] [1275] [1276] [1277] [1278] [1279] [1280] [1281] [1282] [1283] [1284] [1285] [1286] [1287] [1288] [1289] [1290] [1291] [1292] [1293] [1294] [1295] [1296] [1297] [1298] [1299] [1300] [1301] [1302] [1303] [1304] [1305] [1306] [1307] [1308] [1309] [1310] [1311] [1312] [1313] [1314] [1315] [1316] [1317] [1318] [1319] [1320] [1321] [1322] [1323] [1324] [1325] [1326] [1327] [1328] [1329] [1330] [1331] [1332] [1333] [1334] [1335] [1336] [1337] [1338] [1339] [1340] [1341] [1342] [1343] [1344] [1345] [1346] [1347] [1348] [1349] [1350] [1351] [1352] [1353] [1354] [1355] [1356] [1357] [1358] [1359] [1360] [1361] [1362] [1363] [1364] [1365] [1366] [1367] [1368] [1369] [1370] [1371] [1372] [1373] [1374] [1375] [1376] [1377] [1378] [1379] [1380] [1381] [1382] [1383] [1384] [1385] [1386] [1387] [1388] [1389] [1390] [1391] [1392] [1393] [1394] [1395] [1396] [1397] [1398] [1399] [1400] [1401] [1402] [1403] [1404] [1405] [1406] [1407] [1408] [1409] [1410] [1411] [1412] [1413] [1414] [1415] [1416] [1417] [1418] [1419] [1420] [1421] [1422] [1423] [1424] [1425] [1426] [1427] [1428] [1429] [1430] [1431] [1432] [1433] [1434] [1435] [1436] [1437] [1438] [1439] [1440] [1441] [1442] [1443] [1444] [1445] [1446] [1447] [1448] [1449] [1450] [1451] [1452] [1453] [1454] [1455] [1456] [1457] [1458] [1459] [1460] [1461] [1462] [1463] [1464] [1465] [1466] [1467] [1468] [1469] [1470] [1471] [1472] [1473] [1474] [1475] [1476] [1477] [1478] [1479] [1480] [1481] [1482] [1483] [1484] [1485] [1486] [1487] [1488] [1489] [1490] [1491] [1492] [1493] [1494] [1495] [1496] [1497] [1498] [1499] [1500] [1501] [1502] [1503] [1504] [1505] [1506] [1507] [1508] [1509] [1510] [1511] [1512] [1513] [1514] [1515] [1516] [1517] [1518] [1519] [1520] [1521] [1522] [1523] [1524] [1525] [1526] [1527] [1528] [1529] [1530] [1531] [1532] [1533] [1534] [1535] [1536] [1537] [1538] [1539] [1540] [1541] [1542] [1543] [1544] [1545] [1546] [1547] [1548] [1549] [1550] [1551] [1552] [1553] [1554] [1555] [1556] [1557] [1558] [1559] [1560] [1561] [1562] [1563] [1564] [1565] [1566] [1567] [1568] [1569] [1570] [1571] [1572] [1573] [1574] [1575] [1576] [1577] [1578] [1579] [1580] [1581] [1582] [1583] [1584] [1585] [1586] [1587] [1588] [1589] [1590] [1591] [1592] [1593] [1594] [1595] [1596] [1597] [1598] [1599] [1600] [1601] [1602] [1603] [1604] [1605] [1606] [1607] [1608] [1609] [1610] [1611] [1612] [1613] [1614] [1615] [1616] [1617] [1618] [1619] [1620] [1621] [1622] [1623] [1624] [1625] [1626] [1627] [1628] [1629] [1630] [1631] [1632] [1633] [1634] [1635] [1636] [1637] [1638] [1639] [1640] [1641] [1642] [1643] [1644] [1645] [1646] [1647] [1648] [1649] [1650] [1651] [1652] [1653] [1654] [1655] [1656] [1657] [1658] [1659] [1660] [1661] [1662] [1663] [1664] [1665] [1666] [1667] [1668] [1669] [1670] [1671] [1672] [1673] [1674] [1675] [1676] [1677] [1678] [1679] [1680] [1681] [1682] [1683] [1684] [1685] [1686] [1687] [1688] [1689] [1690] [1691] [1692] [1693] [1694] [1695] [1696] [1697] [1698] [1699] [1700] [1701] [1702] [1703] [1704] [1705] [1706] [1707] [1708] [1709] [1710] [1711] [1712] [1713] [1714] [1715] [1716] [1717] [1718] [1719] [1720] [1721] [1722] [1723] [1724] [1725] [1726] [1727] [1728] [1729] [1730] [1731] [1732] [1733] [1734] [1735] [1736] [1737] [1738] [1739] [1740] [1741] [1742] [1743] [1744] [1745] [1746] [1747] [1748] [1749] [1750] [1751] [1752] [1753] [1754] [1755] [1756] [1757] [1758] [1759] [1760] [1761] [1762] [1763] [1764] [1765] [1766] [1767] [1768] [1769] [1770] [1771] [1772] [1773] [1774] [1775] [1776] [1777] [1778] [1779] [1780] [1781] [1782] [1783] [1784] [1785] [1786] [1787] [1788] [1789] [1790] [1791] [1792] [1793] [1794] [1795] [1796] [1797] [1798] [1799] [1800] [1801] [1802] [1803] [1804] [1805] [1806] [1807] [1808] [1809] [1810] [1811] [1812] [1813] [1814] [1815] [1816] [1817] [1818] [1819] [1820] [1821] [1822] [1823] [1824] [1825] [1826] [1827] [1828] [1829] [1830] [1831] [1832] [1833] [1834] [1835] [1836] [1837] [1838] [1839] [1840] [1841] [1842] [1843] [1844] [1845] [1846] [1847] [1848] [1849] [1850] [1851] [1852] [1853] [1854] [1855] [1856] [1857] [1858] [1859] [1860] [1861] [1862] [1863] [1864] [1865] [1866] [1867] [1868] [1869] [1870] [1871] [1872] [1873] [1874] [1875] [1876] [1877] [1878] [1879] [1880] [1881] [1882] [1883] [1884] [1885] [1886] [1887] [1888] [1889] [1890] [1891] [1892] [1893] [1894] [1895] [1896] [1897] [1898] [1899] [1900] [1901] [1902] [1903] [1904] [1905] [1906] [1907] [1908] [1909] [1910] [1911] [1912] [1913] [1914] [1915] [1916] [1917] [1918] [1919] [1920] [1921] [1922] [1923] [1924] [1925] [1926] [1927] [1928] [1929] [1930] [1931] [1932] [1933] [1934] [1935] [1936] [1937] [1938] [1939] [1940] [1941] [1942] [1943] [1944] [1945] [1946] [1947] [1948] [1949] [1950] [1951] [1952] [1953] [1954] [1955] [1956] [1957] [1958] [1959] [1960] [1961] [1962] [1963] [1964] [1965] [1966] [1967] [1968] [1969] [1970] [1971] [1972] [1973] [1974] [1975] [1976] [1977] [1978] [1979] [1980] [1981] [1982] [1983] [1984] [1985] [1986] [1987] [1988] [1989] [1990] [1991] [1992] [1993] [1994] [1995] [1996] [1997] [1998] [1999] [2000] [2001] [2002] [2003] [2004] [2005] [2006] [2007] [2008] [2009] [2010] [2011] [2012] [2013] [2014] [2015] [2016] [2017] [2018] [2019] [2020] [2021] [2022] [2023] [2024] [2025] [2026] [2027] [2028] [2029] [2030] [2031] [2032] [2033] [2034] [2035] [2036] [2037] [2038] [2039] [2040] [2041] [2042] [2043] [2044] [2045] [2046] [2047] [2048] [2049] [2050] [2051] [2052] [2053] [2054] [2055] [2056] [2057] [2058] [2059] [2060] [2061] [2062] [2063] [2064] [2065] [2066] [2067] [2068] [2069] [2070] [2071] [2072] [2073] [2074] [2075] [2076] [2077] [2078] [2079] [2080] [2081] [2082] [2083] [2084] [2085] [2086] [2087] [2088] [2089] [2090] [2091] [2092] [2093] [2094] [2095] [2096] [2097] [2098] [2099] [2100] [2101] [2102] [2103] [2104] [2105] [2106] [2107] [2108] [2109] [2110] [2111] [2112] [2113] [2114] [2115] [2116] [2117] [2118] [2119] [2120] [2121] [2122] [2123] [2124] [2125] [2126] [2127] [2128] [2129] [2130] [2131]

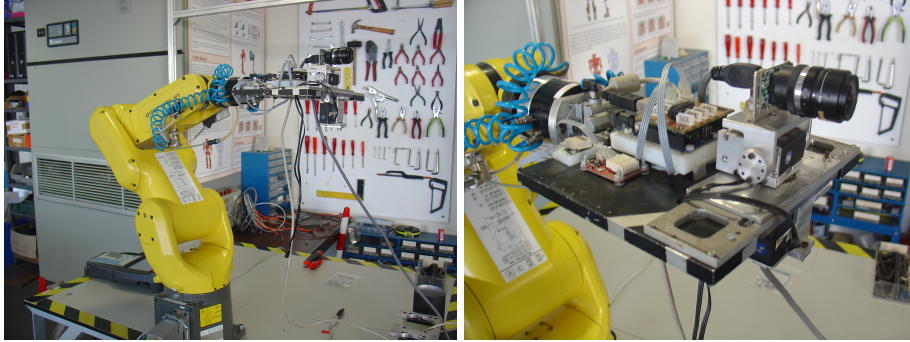


Fig. 1. Experimental setup (left) and a detailed view of the components (right)

[REDACTED]

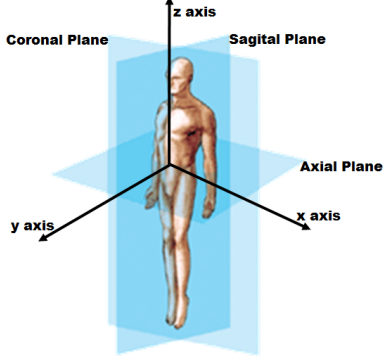
[REDACTED]

[REDACTED]

[REDACTED]

Regarding the measurements, we just need to add as many as we need and then relate them to the state variable matrix. In this case, we have a defined number of outputs from the sensors, which are the angular position and velocities

for the three axis and a variable number of outputs from the camera, resulting in equation (5). It is possible to obtain more than one measurement from the image, therefore we present formula (5) using  $n$  measurements. In these experiments, the camera was aligned with the y axis of the sensor, as we can observe in Fig.2.

$$\begin{bmatrix} \theta_{xS_k} \\ \theta_{yS_k} \\ \theta_{zS_k} \\ \dot{\theta}_{xS_k} \\ \dot{\theta}_{yS_k} \\ \dot{\theta}_{zS_k} \\ \theta_{yC_{k1}} \\ \dot{\theta}_{yC_{k1}} \\ \theta_{yC_{k2}} \\ \dot{\theta}_{yC_{k2}} \\ \dots \\ \theta_{yC_{kn}} \\ \dot{\theta}_{yC_{kn}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \theta_{xk} \\ \theta_{yk} \\ \theta_{zk} \\ \dot{\theta}_{xk} \\ \dot{\theta}_{yk} \\ \dot{\theta}_{zk} \end{bmatrix} \quad (5)$$


**Fig. 2.** Axis Definition from humanoid point of view

The subscript S stands for values measured from the inertial sensor  
The subscript C stands for values measured from the camera

## 4.2 Visual Tracking

When it comes to visual tracking, one of the most common and intuitive ways to analyze an image is to perceive certain image regions (blobs) that can describe an object which the robot can identify. However, it is not always possible to use these global methods, since there aren't always images simple enough to apply this method to. As a result, there is the need to find a method which may be used in more diverse situations.

An alternative way to work with an image is to try to extract some points (features) that may show special properties such as being invariant to scale and position. These are known as local features since they are calculated in key points instead of global methods as those based in blob analysis. There are many different approaches for these local descriptors, and for the work in this paper the features of choice was SURF.

The idea of this method is to keep track of some relevant features in a given image and extract information regarding the transformation between sets of features, frame by frame.

Features are associated to pixels and are most relevant as a group. After finding the SURF features with a specific detection algorithm [6], it is necessary to extract them [6, 7] (extracted features are known as descriptors).

Descriptors save the information relative to a point (in the case of SURF a  $n \times 128$  vector, where  $n$  is the number of features and 128 is used to describe the quality of the point in relation to its surroundings [6]). Having the extracted features, we need to compare the features [8, 9] existing in  $frame_i$  and  $frame_{i-1}$  in order to find the ones that match, i.e., that exist in both frames. The next step consists of calculating the transformation [10, 11] that occurred from  $frame_{i-1}$  to  $frame_i$ , and extract the rotation component.

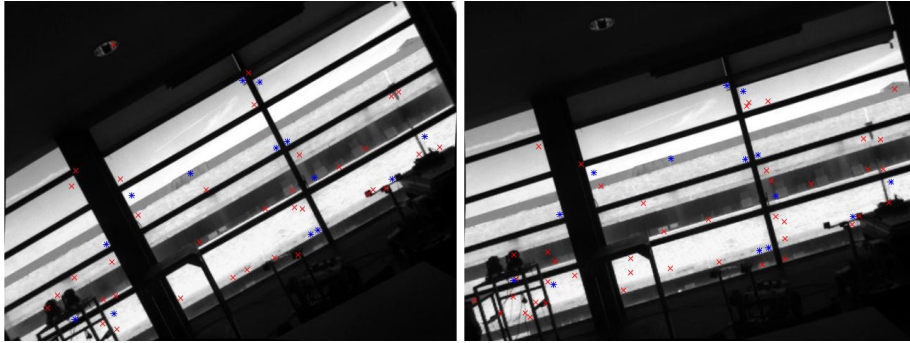
This extracted rotation ( $\Delta\theta_{B_i}$ ) will be the base value to calculate the orientation of the camera. This method allows to obtain a value for  $\theta$  and  $\dot{\theta}$ , (6) and (7), yielding for each frame, a single measurement of angular position and velocity.

$$\dot{\theta}_i = \frac{\Delta\theta_{B_i}}{t_i - t_{i-1}} \quad (6)$$

$$\theta_i = \theta_{i-1} + \dot{\theta}_i(t_i - t_{i-1}) \quad (7)$$

We can easily deduce from equation (7) that  $\theta_i = \theta_{i-1} + \Delta\theta_{B_i}$ . This implies that if the time of acquisition of each frame is unknown, we can still know the orientation of the camera, although we can't perceive its angular velocity. If the time of acquisition is known, we can obtain both the angular position and velocity (which is the best case scenario).

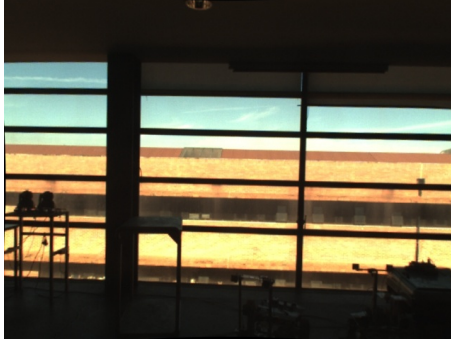
In Fig. 3 and Fig. 4 we can observe two consecutive frames with a portion of the features found (red cross) and features matched (blue asterisk). For visualization purposes, only a fraction (about 1/10) of the actual features is represented.



**Fig. 3.** Detected and Matched Features - frame 32 - first experiment      **Fig. 4.** Detected and Matched Features - frame 33 - first experiment

## 5 Results

The experiment consists of the rotation of the set of camera and inertial sensors around the  $y$ -axis describing a semi-circle with  $r = 150mm$  and  $\alpha \in [0; \pi]$ , using two different backgrounds and illuminations. The *base image frame* (the image frame with  $\alpha = \frac{\pi}{2}$ ) from each experiment is presented in Fig. 5 and Fig. 6. We can observe that experiment one was done under much brighter illumination conditions and has more suitable objects for blob extraction, however blob extraction was not applied in any of the experiments, being used only the feature extraction method to determine  $\theta$  and  $\dot{\theta}$ , thus we can compare equal sets of data that went under the same calculations.



**Fig. 5.** Base frame from experiment one (good light)



**Fig. 6.** Base frame from experiment two (poor light)

The results of the experiments are shown in table 1. These are the comparison between the values obtained from the algorithms and the values obtained from the FANUC robot, using formula (8), which essentially describes the average of the absolute difference between two sets of data:

$$res = \frac{\sum_{t=1}^n |F_t - d_t|}{n} \quad (8)$$

$res$  mean error (displayed on tables)  
 $F_t$  Ground truth from FANUC at instant  $t$   
 $d_t$  data measured or predicted  
 $n$  number of measurements

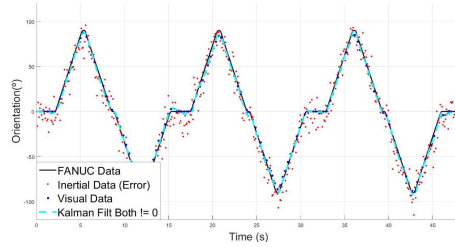
The raw data obtained from the experiment is the one that comes from the sensors, in which we get  $\theta_x, \theta_y, \theta_z, \dot{\theta}_x, \dot{\theta}_y$  and  $\dot{\theta}_z$ . The ground truth data (obtained from the FANUC) is only  $\theta_x, \theta_y$  and  $\theta_z$ . The experiment was performed in such a way that there was only rotation around  $y$ , which will simplify the analysis and will prove the concept for 3D rotation. Therefore, we will need to compare the rotation obtained from the FANUC in  $y$  axis to the rotations obtained from the sensors, camera and merged data.

	Exp1	Exp2	Exp1 (noise)	Exp2 (noise)
Inertial	1.58°	2.80°	8.82°	8.73°
Visual	2.26°	4.68°	2.26°	4.68°
Inertial Kalman ( $\ddot{\theta} = 0$ )	1.56°	2.52°	5.03°	4.35°
Visual Kalman ( $\ddot{\theta} = 0$ )	2.42°	6.16°	2.43°	3.80°
Both Kalman ( $\ddot{\theta} = 0$ )	1.09°	2.43°	2.15°	2.47°
Inertial Kalman ( $\ddot{\theta} \neq 0$ )	1.49°	2.57°	4.86°	3.87°
Visual Kalman ( $\ddot{\theta} \neq 0$ )	1.96°	3.79°	1.99°	2.86°
Both Kalman ( $\ddot{\theta} \neq 0$ )	1.00°	2.49°	2.08°	2.37°

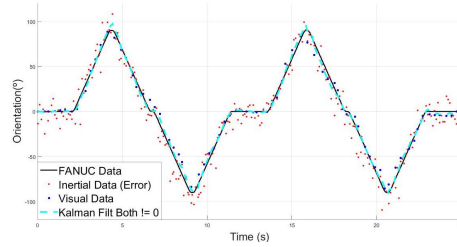
**Table 1.** Results from two different experiments without and with noise added to the measurements. The numbers represent the mean error when compared to the ground truth provided by the FANUC robot.

In table 1, "*Inertial*" is the comparison between the FANUC robot and sensor data (raw), "*Visual*" is the average of the comparisons between the data obtained from the visual data (one or multiple feature tracking) and the ground truth, "*Inertial and Visual Kalman*" is the inertial and visual data processed by the Kalman Filter (no merging), "*Both Kalman*" is the merged data submitted to the Kalman process, whilst  $\ddot{\theta} = 0$  and  $\ddot{\theta} \neq 0$  refers to the comparison of data submitted to Kalman Filter process with  $\ddot{\theta}_k = 0$  and  $\ddot{\theta}_k = \frac{\dot{\theta}_{k-1} - \dot{\theta}_{k-2}}{t_{k-1} - t_{k-2}}$  respectively.

Fig. 7 and Fig. 8 present the the ground truth, inertial, visual and combined data in **Exp1 (noise)** and **Exp2 (noise)** where the Kalman filter including both inertial and visual performed nearly as good as the ground truth.



**Fig. 7.** All data from **Exp1 (Noise)**



**Fig. 8.** All data from **Exp2 (Noise)**

As we can verify in table 1, the inertial data obtained in **Exp1** is very reliable, which almost excludes the need to use other data in order to improve it, since we may be actually distorting its good results. If we apply the inertial data into a Kalman Filter, it improves, but it in a negligible manner. This may be explained by the fact that the model used in the Kalman Filter is not describing the movement of the robot but rather the general laws of motion regarding angular displacement. The visual data is worst than the inertial data, however, when

we apply this data (only) into a Kalman Filter, the output of the filter has a great increase in accuracy. The results are better when we calculate  $\dot{\theta}$ . As it was expected, joining inertial and visual data into a single Kalman Filter improves the accuracy of result, surpassing the individual accuracy of each.

In experiment **Exp2** the results are not so promising. The sensors data is slightly poorer and the visual data is much worst, as it was expected, since the captured images weren't as good as in the first experiment. The deficiency in illumination greatly influences the experiment (Fig. 5 and Fig. 6). Despite that, the conclusions are similar to the ones obtained in **Exp1**.

In order to understand how this technique behaves in less accurate data, noise was added to the sensor data (to each value was added a normal distributed random number in the range  $[-10^\circ; 10^\circ]$ ) and the exact same calculations were repeated.

At this point, the sensor data is highly inaccurate and no good conclusion may be taken from it in order to perceive the robot orientation.

The visual results may be better, but aren't still accurate enough to use due to a lack of measurements in time (each frame takes around 0.3s in order to be taken and processed as the sensor data is more than 3 times faster). In experiment one, we can see that the usage of vision data improves greatly the accuracy of the Kalman Filter data. Calculating the acceleration during the Kalman Filter process improves slightly the output results.

Remember that the model used in Kalman Filter plays a role in guessing the state variable matrix value. The model used does not predict the movement of the robot, since we don't know what the robot will do, but even so, it can greatly improve the results when a large error occurs, thus proving the power of this tool.

The same conclusion may be taken from **Exp2 with noise**. In this case, since the visual data isn't as good as in **Exp1 with noise**, the output from the Kalman Filter isn't as good either.

It is also important to notice that "*Visual*" and "*Visual Kalman*" have the same rate of data acquisition (about 3 Hz), whilst all the others have the same acquisition rate as "*Inertial*" (about 10 Hz). This is relevant because, even though "*Visual*" data may have a smaller error, it also has less measurements per second than "*Both Kalman*", which may affect the response of the humanoid.

## 6 Conclusions and Future Perspectives

This paper studies the effect of merging visual and inertial data with a Kalman Filter to measure a robot angular position and velocity. The trials were successful, proving that it is possible to use different sources of measurements in order to merge and improve them into an overall set of state variables that describe the behavior of the object of study, as shown in chapter 5. The Kalman Filter works better when we try to deduce the angular acceleration at every iteration, however, not doing so does not present itself as a big loss in accuracy.

When the data is highly unreliable (inertial data with error), we can use subsets of external data (visual data) that isn't fully reliable by itself, but may help in filtering the noise in the initial data.

In conclusion, this approach was validated by the results and the next step is to try to implement this method in a real-life situation with real-time calculations. There is a problem that must be solved in order to accomplish this, which is the synchronization of the inertial and visual data when being processed by the Kalman Filter. In this work, all the image related calculations were made and then fed into the filter. In real-time experiments, the time that the image needs in order to be processed may be a challenge when trying to implement the filter. Some modifications to the system may be of need.

## References

1. Vítor Santos, Rui Moreira, and Filipe Silva. Mechatronic design of a new humanoid robot with hybrid parallel actuation. *International Journal of Advanced Robotic Systems*, 9, 2012.
2. A. I. Comport, E. Malis, and P. Rives. Accurate quadrifocal tracking for robust 3D visual odometry. In *Proceedings - IEEE International Conference on Robotics and Automation*, 2007.
3. Stephan Weiss, Markus W. Achtelik, Simon Lynen, Margarita Chli, and Roland Siegwart. Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments. In *Proceedings - IEEE International Conference on Robotics and Automation*, 2012.
4. Konstantine Tsotsos, Alberto Pretto, and Stefano Soatto. Visual-inertial ego-motion estimation for humanoid platforms. In *IEEE-RAS International Conference on Humanoid Robots*, 2012.
5. Morgan Quigley, Ken Conley, Brian Gerkey, Josh FAust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Mg. ROS: an open-source Robot Operating System. *Icra*, 3(Figure 1), 2009.
6. Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3), 2008.
7. Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*, volume 1. 2008.
8. David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2), 2004.
9. Marius Muja and David G Lowe. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. *International Conference on Computer Vision Theory and Applications (VISAPP '09)*, 2009.
10. Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. 2004.
11. Philip H. S. Torr and Andrew Zisserman. MLESAC: A New Robust Estimator with Application to Estimating Image Geometry. *Computer Vision and Image Understanding*, 78(1), 2000.