前言

很多朋友私信问我对机器人和人工智能感兴趣，该怎么展开学习。最近稍微有点空，我写写我的看法。

两年前，我在知乎回答如何定义「机器人」？ - YY硕的回答中试图给机器人做出一个比较仔细的定义，我觉得机器人和人工智能最大的区别在于是否要和物理世界进行交互。今年初在另一篇知乎回答里对机器人或人工智能的研究会帮助我们更好的了解人类自己吗？ - YY硕的回答我说到传感器是和物理世界交互的基础。后来，我又在知乎回答有哪些与控制、机器人等相关的 quotes？ - YY硕的回答中提到莫拉维克悖论（Moravec's paradox），谈到了机器人学里公认的难题是在物理世界中实现类人的活动能力。

把之前的回答再翻出来是为了支持以下观点：机器人学的核心问题是做好和物理世界的交互。现在主流的机器人学分支里，处理与物理世界的交互的学科分为三类：传感器和处理算法（激光雷达，多目视觉，融合算法）；多刚体系统动力学控制（工业机器人动力学控制和接触力控制）；机器人自主移动（locomotion不知道该怎么翻译，轮式、足式、飞行等移动机器人的研究）。我建议对机器人学有兴趣的同学着重在这几个问题上面。

另外，根据世界第一的机器人教育机构卡耐基梅隆大学的机器人学博士的课程分类方式（

），机器人学有四个核心领域：

1. 感知。视觉传感器、图像传感器、触觉和力传感器、惯导等。
2. 认知。人工智能、知识表达、规划、任务调度、机器学习等。
3. 行为。运动学、动力学、控制、manipulation和locomotion等。
4. 数学基础。最优估计、微分几何、计算几何、运筹学等。

结合卡耐基梅隆大学的核心课程要求，我觉得我定义的机器人学核心问题算是基本没跑偏的。本文后面谈到的机器人项目都是以上述观点和课程要求为基础。

一些可能有争议性的观点：

1. 机器人学是富人的活动。虽然工业越来越发达，但好的开发板和电机还是非常贵。如果要下定决心学习机器人学并且做出实物，你必须找到做实物出来的资金。要么是自己花钱，要么就得找学校的机器人社团，或者找什么愿意资助年轻人学习的贵人。另外现在没有任何一本完整的书可以教你怎么造一个四旋翼空中机器人或者大狗机器人，你需要参考十几本不同的教科书，这些书不管中文版还是英文版都很贵。

2. 机器人学是屠龙之术。这话是Ninebot创始人说的。虽然最近几年，平衡车、扫地机器人、多旋翼飞行器让机器人学开始进入人们的生活，但是可行的商业应用还是很少，而且已有的机器人和理论都还很难解决好与物理世界交互这件事情。所以一定要确保自己在机器人学这个道路上同时练好了能去其他行当吃饭的技能，比如编程、机械设计和硬件设计。也要做好心理准备，接受自己有可能在学会屠龙之术以后几年都造不出对社会有用的东西的事实。

3. 基于上述所说的观点，如果是已经工作之后才想要学习机器人的话，可能已经太迟了，因为很可能**兴趣战胜不了客观限制因素**。如果作为兴趣去学习，只能学到做巡线小车和舵机机械臂什么的，可能也满足不了中二病的创造欲。

个人认为机器人学是一个艰苦的道路，想要成为一个独挡一面的机器人工程师需要多年理论和实践的同步训练。理论学习和动手实践的过程还要互相排好时间表，在做某个实践项目的同时去学习最相关理论往往可以达到事半功倍的效果，但是同时那些不太相关的理论会看起来非常枯燥，因此如何妥善安排自己的实践项目也是很重要的事情。

这篇文章里我计划介绍一个电子工程、机械工程和计算机专业学生从大学一年级到研究生二年级的机器人学习计划，基本是我对自己过去学习方式的一个总结。按照这个方法来学习，能够成为一个能力全面，但是稍微偏软件一点的机器人工程师。这个六年的学习计划，估算下来，全年中每天在上课和完成课程要求之外要投入学习时间6-8个小时，这些时间一方面用于阅读课程知识的英文教材，一方面用于阅读其他学科的教材，一方面用于实践项目。

机器人工程师在大型项目里的定位类似于飞机系统里的总体设计师。和机械工程师、硬件工程师、软件工程师、算法工程师、控制工程师比起来，机器人工程师参与某个具体技术的时间较少，但是能够听得懂所有工程师说的话，能够作为不同模块间的协调人，带领整个团队去攻坚。当然如果机器人工程师能够在一个领域达到那个领域的工程师的优秀水平，肯定更好。

由于时间仓促，再加上个人水平有限，文章中如有纰漏和错误，恭请读者指出，谢谢。如果同学还有什么想知道的内容和教材，也欢迎留言交流。

大一

刚上大一，你的机器人生涯开始了。先看看学校的校园网能不能翻墙，不能的话自己去买个一年一百多块钱的VPN，先确保自己能上Google，不要心疼VPN的钱，这能让你在之后的职业生涯里节省上万块钱。然后去注册一个gmail账号，再注册stackoverflow账号，再注册github的账号，再注册CSDN账号，注册完登录上去逛逛，暂时先不要问为什么。

英语水平一定程度上会是机器人工程师水平的限制因素，英语是同学们在大学最该努力学的一门课，而且不止要把它当成课，要当成一种技能，当成生活的一部分。当你开始努力学习一些高级的机器人知识以后，有可能会非常难以找到中文的参考资料，这个时候如果啃不下英文的资料，进步速度和眼界就会受到很大影响。因此大一的时候要多看看红宝书，看看美剧。

不管是什么专业背景的同学，大学一年级一定要上好的课是微积分和线性代数。线性代数的重要性需要特别强调。一般来说，**优秀的工程师和科学家在职业生涯中要学至少五次线性代数**，大一学一遍、学凸优化的时候学一遍、学线性系统的时候学一遍、学机器学习的时候学一遍……如果在第一遍学的时候就看到对的书，刷到对的题，那么以后的学习会轻松很多。

网上有很多对于如何学好线性代数的讨论，比如知乎问题如何理解线性代数？ - 数学学习。Matrix67大神的文章随记：我们需要怎样的数学教育？也很有启发。我个人对学习线性代数的建议是两本书，一本叫做《Linear Algebra Done Right》，另一本叫做《Linear Algebra Done Wrong》（ math.brown.edu/~treil/p... ），我比较喜欢的是Done Wrong这本书，第一它是免费的，第二只需要读前6章两百页就够了，第三它页边距很大，打印出来有很多空白做练习题。另外一个较好的教材是麻省理工公开课：线性代数。不论如何，学线性代数一定要用国外的教材，千万不要用国内的教材。啃英文书很累，但是考虑到之后还要啃更多的英文书，线性代数已经算是很入门的了，一定要啃下来，同时还要刷足够多的课后题。

学完线性代数以后，一个自然而然的问题就是怎么能用计算机自己去计算矩阵的乘法、向量的乘法、向量的内积。因此引入了编程的学习。

不管同学的专业是什么，一定要在大学一年级尽早开始学编程。至于用哪种语言开始学习编程，我推荐Python，比较好的教材是麻省理工学院公开课：计算机科学及编程导论，比较好的Python开发学习环境是Anacoda (continuum.io/downloads)。熟悉Python以后，同学就可以开始玩玩Python的数值计算包Numpy，这个时候线性代数题基本上也刷的差不多，可以通过Numpy帮助自己解决线性代数问题了。

对任何人来说，Python是一把瑞士军刀，你可以用他干很多东西，比如自动回复邮件、自动收集信息。但是真要去造机器人，合适的工具并不是瑞士军刀，而是C/C++这样简单粗暴的锤子和螺丝刀般的工具。在学习Python学到一定程度的时候（比如你听说有一种叫做cython的东西），最

好开始学习C，而且要强迫自己练习用C的一维数组和指针来实现矩阵的加法、乘法、求逆等操作。之所以有高级的Python或者Java（不要问我Java哪里高级了）这些语言以后我们还需要去学C，是因为机器人上常用的不是完整的电脑，而是计算量有局限的嵌入式系统，嵌入式系统开发基本只能用C或者更低级的语言。

学习C我个人入门用的是清华大学出版的《C++语言程序设计》。虽然这个书标题是C++，但其实没什么太大问题。不过国内的C语言教材都有个巨大的问题是不引导学生去用Linux。近年来更好的一个教材是songjinshan.com/akabook...，这个网站的教材非常好，因为他教育学生用Linux环境作为程序编译的环境，而且还引入了一些计算机体系结构的介绍。

IT行业的程序员都会争论高级语言和低级语言哪个好，Linux和Windows哪个好，而对于机器人工程师来说，从现在到可预见的未来里，C是最好的语言，Linux是最好的操作系统，这都毋庸置疑。甚至对于Linux的发行版该选哪个，我们都是很少有质疑的：Ubuntu（The leading OS for PC, tablet, phone and cloud）。原因是机器人操作系统ROS（ROS.org | Powering the world's robots）是基于Ubuntu开发的，因此在Ubuntu上运行最稳定。注意Ubuntu出了一个中文版叫做Kylin，个人感觉比较坑，建议大家不要装中文版。Ubuntu 作为一个开源操作系统，总是在快速迭代，2016年8月比较稳定的版本是14.04和16.04，建议同学安装14.04。

当你把C学得差不多，开始要学写包含多个头文件的程序时，一定要同时学习makefile的知识。这时候要上网去搜“Makefile详解”（Makefile详解（超级好）_mingw吧）这篇文章看。

我自己在大学一年级的时候还学习了HTML和Javascript，到大一结束的时候已经能够熟练用Javascript手写一些动态页面。我个人觉得HTML和Javascript也是机器人工程师必备的技术，而不只是软件工程师的玩具。这是因为web技术实际上已经渗透到了编程的方方面面，比如json开始是Javascript里的一种object定义的方式，但现在已经成为了一种很标准的数据交互、参数配置的格式。另外AJAX能够帮助初学者理解一定的网络技术原理，而网络技术也是机器人工程师必备的技能。再者，制作GUI（图形用户界面）是常规debug的办法，而近年来一个流行的趋势是用webkit嵌入程序用HTML和Javascript作为图形界面的后端，而在机器人操作系统ROS（ROS.org | Powering the world's robots）里，通过rosbridge可以非常方便地把机器人程序的数据传递到websocket上，这句话看不懂没关系，反正你知道学学HTML和Javascript很重要就是了。更重要的是，HTML文档背后的DOM (Document Object Model)深刻地体现了面向对象的思想。大学中的面向对象程序设计一般都讲C++，在我看来应该讲HTML和Javascript。这一点不细说了，如果同学们去学习HTML和Javascript，自然会体会到。学习HTML和Javascript比较好的资料是w3schools.com/，把网站左侧的“Learn HTML”、“Learn CSS”、“Learn Javascript”和“Learn JQuery”学完就行，别的部分还有很多花哨的技术，没有必要去学了。为了培养自己对Javascript的兴趣，可以上three.js / examples跪着看看热闹。

以上介绍的这些知识点、书和资料应该在大一期间就全部看完，然后利用大一的暑假好好巩固这些知识。比如开始用HTML和Javascript做一个自己的个人主页，刷一刷编程的题目，学用Python的奇技淫巧（比方说做一个自己的个人主页）。另外还可以抽时间学学数学知识，比如开始看看代数和离散数学。我大一的时候看到了两篇文章，认识到了数学的重要性，一个是MIT的CV大牛林达华写的[转]MIT牛人解说数学体系，另一个是前Goolge研究员吴军博士写的《数学之美》（数学之美 (豆瓣)）。当时林达华还在MIT读博士，而《数学之美》还没有成书。两篇文章看完以后我感觉自己整个人对数学的认识上了一个新的层次，此后一直在注意提高自己的数学水平，几年下来觉得收益很大。在之后的介绍里我还会多次强调需要学的数学知识和对应的教材。

大二

上大二的时候，你已经会了基本的编程知识和基本的数学知识。大二这一年应该投入在嵌入式系统的学习中，同时继续拓展自己多方面的能力。学校的机器人社团，比如做Robocon，RoboMasters的团队应该在招新了，赶快去加入，有了学长学姐的指导以及同辈朋友的鞭策，应该会进步的快一点。

大二应该掌握的技能：Solidworks画基本的机械图，基本的数字电路知识、数模转换，51单片机、AVR单片机、STM32单片机原理，UART、SPI、I2C、CAN等协议的原理和数据收发，STM32开发板的使用，电机转动和驱动的原理，PID的原理，调试四轮机器人底盘的移动，基本的传感器如陀螺仪、码盘、红外线、超声波的原理和读取方式，网络知识如配置IP配置路由器等，微电子焊接，金工技术。我在知乎回答如果程序员每天都浅尝辄止地学一些不同的新技术，长久以往，人会变成什么样子呢？ - YY硕的回答里谈过机器人工程师需要的技能数量是IT行业全栈工程师技能数量的三倍以上，这些技能的基础都应该在大二开始积累。

如果同学们的专业是机械工程相关，那么大二的时候要深入学习solidworks做图，买机械加工手册学习各种机械的奇技淫巧。你的专业知识还不足以让你进行缜密的受力分析，不过你可以尽量多做一些机械结构出来感受它们的乐趣。

如果同学们的专业是硬件、电路相关，那么大二的时候要深入学习Altium Designer做图、制板、焊板。你要从现在开始，就给自己积累一个工作记录，可以就是简单的txt文件，记录你做过所有板子的bug、解决方案、学到的原理图、PCB layout的注意事项等等。积累很多年以后，你的这个工作记录会值很多钱。

学习这些技能的最好的方式，就是参加自己学校机器人社团的训练和方案设计。一般来说，学校的机器人社团招新之后会有训练和测试，让新人分组去做机器人，这个过程中如果愿意努力学，提高得会很快。如果你所在的大学是机器人比赛强校，比如西安交通大学，电子科技大学，哈尔滨工业大学，华中科技大学，东北大学等等（排名不分先后，没有提到你们学校名字的话我表示抱歉），那么你很幸运，你们学校的机器人社团有很好的积淀，有很多资源可以帮助你学习。基

本上只要天天泡实验室，保证自己每天只睡6-7个小时（但还是要多去跑步、游泳保持身体健康），勤于向学长学姐请教，那么一定会提高得很快。

大二阶段特别要强调的是对动手能力的培养，包括机械材料的加工、电路焊接、制作导线和接头、连接路由器、配置网络、做网线等等。机械加工的工具有螺丝刀、锯、钻、锤子、车床、铣床、钻床，进阶选手可以学一下氧焊，这个比较危险，我没尝试过；电路焊接的工具有焊机、焊锡、洗板水、松香、吸锡器；制作导线的工具有剪子、剥线钳、夹头钳、网线钳各种钳；网络配置就是连连路由器插插网线，但是Linux系统下配置网络有时会非常麻烦，一定要多积累这方面的知识，因为将来你造的机器人多半会顶着一个无线路由器跑来跑去，甚至有的机器人上各个模块自己就能组起一个小局域网。这些技能的熟练掌握需要你花很多时间去做真正能用的机器人来练手。

对于该选择造一个怎么样的“真正能用的机器人”练手，最好的选择肯定是机器人比赛中的机器人。如果参加Robocon，你会跟着学长学姐们学着造有人那么高的巨大机械；如果参加RoboMasters，你会学着造比汽车还要灵活的机器人以及快速发弹的机构。其他一些小型的比赛比如飞思卡尔智能车，也是很好的训练，因为飞思卡尔智能车已经发展得很成熟，参加这个比赛的参赛资料就够学一阵子的，学完以后能够获得比较多的机器人技能。

如果没有太多学校机器人社团的资源，同学们还有一些小型的比赛比如挑战杯、大创比赛等等可以选择，以三五个人的小团队参与这些比赛。如果同学所在的学校连这些比赛都不组织大家参与，那就只好自己花钱了。国内开源机器人社区有很多资源可以利用来学习，比如自己买Arduino STEM educational Robot kits Building Platform的各种开发套件做简单的机器人。Arduino的开发环境可能有些人不喜欢，因为它对硬件做了一级封装，如果更希望接触到单片机的本质，可以自己买STM32开发板学习。俗话说，没有什么嵌入式系统是一块STM32实现不了的，如果有，就用两块。STM32是ARM Cortex-M家族中最为广泛应用的一款单片机，在网上也有很多的教材和开发板可供选择。在国内著名的电子论坛STM32/8 分论坛帖子清单 (amoBBS 阿莫电子论坛)上，有很多参考资料，有问题也可以在这里和大家讨论学习。

如果你很想参加机器人比赛，身边也恰好有一些志同道合的小伙伴，但是学校不支持。没关系，来找我，我尽量通过大疆的关系说服你们学校支持你们参加RoboMasters。

虽然你是以机器人比赛为主线在探索机器人技术，但是要时刻记得，机器人比赛给你的理论方面的训练很差，还会让你养成一些坏习惯，比如凡事都希望用一些糙猛快的办法来解决。由于通常整个团队都没有太多的项目管理经验，到比赛前一段时间才会加紧功夫去做机器人，很多时候就会用“山寨”的办法去处理机器人的故障。比如说某个承重结构用久了会弯，为了赶比赛的进度，就拿锤子敲直了、再加一条辅助的结构在旁边继续用，而没有细致地去做建模、受力分析，思考是什么原因导致承重结构会变形。再比如说调PID参数就是生调乱改，而没有基于机器人的动力学模型去估测参数的大概范围。

大二的暑假，有可能你跟着学校的机器人队参加了一些机器人比赛。这是一个反思总结的好时机，为什么机器人队取得了这样那样的成绩？整个团队怎样才能更有效率？明年如何继续招新？暑假要把时间花在技术积累上面，这个时候可以回头思考思考之前准备比赛时用糙猛快的办法解决的问题如何能够细致地去解决。

如果大二的暑假没有参加机器人比赛，可以做一个舵机机器人，比如6条腿的蜘蛛，比如码垛机器人。舵机是机器人工程师的好朋友，一定要好好掌握。

另外你其他方面的能力也不能落下。大二结束的时候，你的Linux应该用的很熟练了，除了makefile，你也用起了cmake。你也应该开始理解Github存在的意义，因为你已经上去读了很多别人的代码，你也把自己的一些课程设计和小项目放在了Github上面。另外大二基本上了本专业一些比较难的基础课程，比如自动控制原理、机器学习、概率统计、材料和力学等课程，同学们会看到这些课程里又用到了线性代数和微积分的知识，以及建模的知识。这时候可以把大一的物理、线性代数和微积分再翻出来看看。

同学的学校应该给大二到大三的学生有开设面向对象的程序设计，一般用Java或C++教授。在面向对象的程序课里面，一定要积累3000行左右的代码的开发经验。经典的面向对象程序设计的练习通常是写游戏，比如俄罗斯方块，吃豆人等等，一定要自己能够做到完全手写一个完整的项目出来。

另外你可以开始学习Matlab当中的神器Simulink了。在大二这一年的学习中，你可能在不少课程里多多少少用到了Matlab。假设你已经在我的推荐下喜欢上了Python，你可能会觉得Matlab的计算工具没有比Python强多少；假设你自己在别人的推荐下喜欢上了mathematica（Mathematica 到底有多厉害？ - Wolfram Mathematica），你可能会觉得和Mathematica这种神一样的语言比起来，Matlab弱爆了。但是要注意的是，Matlab最强大的工具是Simulink，通过它你几乎可以仿真一切的物理系统和控制系统。我建议同学可以通过Simulink实现一个倒立摆，然后理解Matlab的强大之处。对此我强烈推荐一个很好的教材（http://ctms.engin.umich.edu/CTMS/index.php?example=Introduction§ion=SimulinkControl），它详细介绍了一些经典的控制系统如何分析以及用Simulink实现。然后我再强烈推荐一个讲Simulink里面一个更加和物理仿真贴近的工具Simscape（Control of an Inverted Pendulum on a Cart）的文章。阅读并实现了这两篇文章里的内容，同学应该会对倒立摆有了比较深刻的认识。倒立摆是机器人学中一个非常重要的模型，因为火箭、导弹、双足机器人、四足机器人，基本都是倒立摆的变形。你自己实现出来的simulink模型一定要存好，以后可能还会再拿出来仔细看。

如果你按照我之前说的方法探索了一些HTML和Javascript的技术，那么JQuery，bootstrap，AngularJS你已经多多少少知道是怎么回事了。web后端的技术，比如PHP和MySQL，也可以了解一下，LAMP要学会怎么配置。如果有同学找你帮忙写小网站，尽量去帮个忙，帮别人做网站是提高自己系统编程能力的好练习。在做网站的过程中你可能还会顺便学一学如何用Photoshop和Illustrator让网站显得更漂亮。这些技能有最好，没空学也没关系。

大三

大三开始的时候，你可以在学校的机器人队里担任重要角色了，或者能够带领一个小团队参加小型的机器人比赛。你在系里甚至院里都小有名气了，可能有的人叫你大神，有的人觉得你技术还不错。但是一定要记得你现在的水平放到别的地方应该不算什么。每年我都往大疆的RoboMasters夏令营招进100个和你现在的水平相当的同学。

大三一定要培养出自己一个人独立造出一个完整机器人的能力，比如一个Robocon水平的机器人，或者一个RoboMasters的战车，或者一个四旋翼飞行器。

Robocon水平的机器人，涉及大量的机械设计，单片机开发，电机驱动的开发，码盘和超声波等传感器的读取，底盘运动学的计算，PID调试，任务调度逻辑的调试，舵机控制。RoboMasters的战车的机械部分简单一些，但是还需要进行云台的控制、发弹系统的控制、功率控制等方面的知识，另外你也可以不搞这些部分，学习学习视觉识别和自动打击，那么就要开始研究OpenCV（OpenCV | OpenCV）了。

四旋翼飞行器的机械部分最简单，但是算法比较复杂。对于大三学生来说，从零开始一步到位写一个稳定的飞控比较困难，因为飞控里面有很多细小的知识点要注意。目前我没有发现什么比较好的书籍推荐，已有的一些关于四旋翼系统的书要么太浅（上来就教你焊电路），要么太深（上来就教你state estimation），听说 @Liu Top的exbot小组在写一个教材，我是非常期待的。学习四旋翼飞行器有下面几个步骤：

第一步：自己调一个小四轴飞起来

现在开源社区的人言必pixhawk，其实我觉得从学习的角度来说，pixhawk太贵，而且不适合学习，我比较推荐的是 首页-第七实验室 这家淘宝店卖的STM32F405飞控，买回来以后自己再随便买个机架（比如大疆F450）、接收机和遥控，就能按照飞控板附带的学习资料、调试软件飞起来。

第二步：看硬件图、读代码

chiplab7的飞控板附带一大堆学习资料，对加速度计、陀螺和磁感计都有很仔细的解释，硬件链路图也很详细。chiplab7淘宝掌柜的又很认真负责，我学用的时候，发现代码有bug和看不懂的

地方，都可以直接找掌柜问。

看完代码以后，对一个飞控系统的基本模块：姿态解算、控制解算、混控输出、遥控器处理、嵌入式处理就很明白了。然而这里面有很多技术是需要另外学习的。除了基本的嵌入式编程以外，还有要把大二大三学的信号处理方面的知识再捡起来看看。因为飞行器在空中有振动，会让加速度计产生噪声，为了把这些噪声去除掉，需要对加速度计给出的信号做低通滤波处理，如何选择滤波器的参数呢？如果滤波滤得太狠，延迟就会比较大，对控制的表现会有影响；如果滤波滤得不够，可能会有一些低频的噪声偶尔会出现，导致加速度计的观测不能用。另外最重要的是要理解姿态解算和控制解算这两块知识。chiplab7的飞控板的代码采用的是最简单的互补滤波算法做为姿态解算模块，然后控制解算是对欧拉角的三个角度做闭环PID控制，基本都是基础的基础了。

第三步：小修小改加深理解

chiplab7的飞控是靠气压计定高的，飞行效果非常奔放。这时候可以淘宝买个20块钱的超声波模块，然后自己写个高度环去稳定飞控的定高表现。

我觉得这个过程至关重要，因为高度控制相对来说是个比较直观理解PID控制的方式，而且chiplab7的飞控加高度控制非常好加。工作量不大，因为改善效果很显著，所以可以让人很有成就感，加深继续学习的乐趣。

第四步：理解核心的数学和控制知识

这一部分大三是肯定来不及学的，但是我还是在这里列出来，因为这些知识你之后都需要慢慢学，我也会在之后不断重复提到这些知识点。

姿态解算和控制解算涉及的知识有：

1. 刚体姿态的表示、运动学方程和动力学方程。主要是对牛顿-欧拉方程的认识和理解、刚体姿态的欧拉角表示法、姿态与角速度的关系等等。

这部分说复杂不复杂，说简单也不简单，我同样是没有找到一本完整的书全都介绍过的，是学了好几个不同的书和论文以后搞明白的。现在看起来是从维基百科入手比较靠谱。

2. 自动控制原理。讲PID的书和文章就多了去了，没有太多复杂的书。

3. 线性估计基本原理。其实就是互补滤波：Reading a IMU Without Kalman: The Complementary Filter 。拿这个关键词百度各种搜就会了。

第五步：重头开始造轮子

知乎著名网友vczh曾经说过，学习要抱着勇于造轮子的心态才能进步。所以在熟悉了别人飞控基础上，可以自己重头造一个飞控的轮子。可以自己从芯片开始重新画一个飞控板，读读STM32的芯片手册、读读各种传感器的芯片手册，自己手画一个飞控的原理图、做PCB layout、制板自己焊元件，全套花不了1000块钱，能够加深很多对硬件的理解。这一部分如果大三没空，也可以不搞了。

制造整个机器人的过程中要特别重视文档的积累。在你大三末期，你可能随着学校的机器人队备战比赛，你可能主力负责一台机器人。你应该自己列一个excel表格，把机器人用了几颗螺丝，几根导线，每个零件的规格是什么，都列出来。这个表格一方面可以用来帮助团队管理机器人的物料，一方面也是你自己的经验技术积累，将来你做的其他机器人可能多多少少都是Robocon、RoboMasters机器人的变形。

另一个积累是建一个自己的buglist，buglist包括什么呢，可以像冷大这样：做控制、机器人等算法工程师是怎样一种体验？ - 冷哲的回答，就简单把一些自己的发生过的问题和最后的解决办法罗列下来。比如说“杜邦线接插位不稳固容易脱开，接好后应该用电工胶布再裹一圈”，“外发给淘宝加工的机械图纸，要特别注意和加工商沟通有没有漏掉一些细节，如沉头螺丝孔，关键的倒角”等等。你也可以帮其他机器人的问题也做这样的记录。buglist可以就是一个简单的文本文件，如果你一直往里面积累自己工作中的记录，等你将来工作了，这个文件可能会值很多钱。

大三的时候学校应该会开设软件工程的课程。不管你是不是这个专业，上不上这门课，都应该主动去听一听，甚至跟着课程的设计作业一起做一做。软件工程我觉得是机器人工程师必须具备的意识，因为一个机器人系统里涉及大量的硬件系统和软件功能，软件的部分往往还会涉及不同的语言、不同的编译环境、不同的开发工具链。几个人合作的话，大家的专业背景、编程习惯都不相同，这就导致不同的代码和模块之间的协议沟通非常复杂，必须尽早用UML和其他软件工程的工具帮助团队理解和互相沟通。

大三的时候学校应该还会开设操作系统原理和嵌入式系统原理的课程，而大二的时候讲过计算机组成原理（所谓的微机原理）。从大三开始同学需要开始体会实时操作系统和非实时操作系统的区别、原理以及使用时需要注意的地方。这是一个比较杂的知识点，我目前没有找到很好的教材去介绍。在STM32上，有freeRTOS，uCOS，Vxworks这么几种实时操作系统；Linux是一种非

实时操作系统，但是可以通过打补丁变成实时操作系统。这些操作系统的细节在机器人开发中都会多多少少被涉及到，同学们可以随时上Google和CSDN去查大神们的介绍。

另外特别重要的一点是机器人系统里的嵌入式平台都有烧坏的可能性，有可能在某个嵌入式Linux平台上面辛辛苦苦写了一个多月代码，这个平台突然烧坏了，代码也就丢了。因此你的机器人如果有嵌入式Linux系统在里面，一定要尽早顶起来路由器，代码定时提交SVN或者git。

大三的暑假你可能会作为学校机器人队的主力去参赛了。备战比赛和参赛是一件磨练心性的事情。我在学生时代体会过和胜利擦肩而过的痛苦，体会过没机会再来一年的遗憾；也在负责大疆RoboMasters比赛的过程中被那些痛苦和遗憾的学生当做发泄的对象，非常有感触。我觉得参加机器人比赛，很努力，然后失败了，是一件让人快速成长的事情。同学如果有机会，一定应该参加至少一届机器人比赛。

大三的暑假你也可以选择来参加大疆的RoboMasters夏令营，关于夏令营大家可以看这个知乎问答了解更多：参加Robomasters 2016夏令营是怎样一种体验? - DJI 大疆创新。每年我们都在全国范围内寻找有一定技术基础的学生，让他们一起分组做一个自动机器人的挑战。这个夏令营，作为组织负责人，不谦虚地说，我觉得应该是全世界范围内最好的技术类夏令营。

大三的暑假有一件很重要的事情就是思考自己大四应该干什么。一般来说，你现在的能力保本校研究生肯定没有问题，当然你也可以选择考其他学校的研究生或者出国留学。虽然说你现在能力已经很全面了，但是你还需要2-3年的时间全面提升自己更多的能力，才能迈向卓越之路。不管是出国还是保研，最重要的目的是给自己争取到未来2-3年能够在一个优秀的环境中安心提升自己，有比较好的学习资源，能够参与到一些不错的项目中去。可能其他有些行当，出国留学始终是比在国内待着更好的选择，但是机器人行业并不是这样。我们国家这两年在机器人方面提高也很快，而且我们国家现在比较有钱。就像我开始说的那样，机器人是富人的活动，现在你在国内也能找到一些很有钱的实验室可以造比较牛逼的机器人。另外国外很多比较强的机器人公司也都在做比较敏感的军方项目，去找实习可能比较受限制。

出国去学机器人学方面的知识你有很多不错的选择，比如世界第一的机器人研究院卡耐基梅隆大学，或者麻省理工学院的CSAIL实验室。北美传统计算机四大名校（麻省理工学院，卡耐基梅隆大学，斯坦福大学，加州大学伯克利分校）里，除了斯坦福大学热火朝天在搞人工智能以外，其他几个学校的机器人研究都很不错。除了四大名校，你还有很多其他的选择，就像我开始说的那样，机器人是富人的活动，如果想接触到最好的机器人资源，你要选择有钱的实验室，而不是有名的实验室。

另外你还需要在大三的尾巴上选定自己将来的细分研究方向，而且开始往这个方向深挖，也就是我在文章开始提到的感知、认知、行为几个方向。当然同时你也不能放松其他方面的知识，尤其是数学基础。我在大三的暑假专门找数学系的同学给我开了个数学小讲座，学习了一点抽象代数的知识，对我后来学习密码学帮助很大。同时我也读了一些拓扑方面的教材（有一本很神奇的书

叫做Topopogy Without Tears topologywithouttears.net...），这样才理解了为什么数学分析要用奇怪的符号去解释一些看起来很浅显的道理。

大三阶段的机器人工程师该学什么基础数学是众说纷纭的，在我看来，你要基本掌握“群是什么”，能够用代数的眼光去证明"det(AB) = det(A)det(B)"，还要能理解“用一张纸就可以变出克莱因瓶”（当然是在四维空间里）。另外，你这个时候也要能够意识到自己需要再学一遍线性代数。

大四

大四开始了，你可以开始深挖自己的研究方向，同时也要开始学一些高级一点的通用技术和理论，这时候你和一般的机械、电子、计算机学生就不太一样了，你虽然也在狂编程，但也在狂学习物理和数学。通用技术包括ROS，simulink，gazebo和Vrep等工具。通用理论包括，再学一遍线性代数，学学凸优化、数值计算、旋转表示法等方面的知识。这些知识你在大四仅仅只能开一个头，因为你的大四要实习、毕业、考研、毕设，你会非常地忙。有些人会在大四进实验室和老师发论文，我个人觉得发论文这件事没必要操之过急。你的整个大学期间应该用在广泛涉猎各种各样的知识上面，而不是深入某一个细小的研究问题。

大四可以开始读一些著名入门书籍，我把这些书不分先后地列出来，你没有必要全部去读，而且每本书先读前几章就够了，能读多少尽量读多少。

1. 概率机器人学，amazon.com/Probabilisti...

2. 凸优化，web.stanford.edu/~boyd/...

3. 线性系统理论，amazon.com/Linear-Syste...

4. Multiple View Geometry in Computer Vision，Multiple View Geometry in Computer Vision

5. 线性估计，amazon.com/Linear-Estim...

6. 《机器学习》，周志华老师的书。

7. An Invitation to 3-D Vision，eecis.udel.edu/~cer/arv...

8. Modern Control Systems，amazon.com/Modern-Contr...

9. Rigid Body Dynamics，authors.library.caltech.edu...。说实话刚体动力学理论我没有找到特别好的书。但是刚体动力学理论很重要。

10. Feedback Systems: An Introduction for Scientists and Engineers，FBSwiki

就像我开始说的那样，这些书，大部分特别贵，还好有一些业界良心的作者放出了他们书的电子版。当然你也可以去一些名字都不能说的网站去找影印版。

在读上面这些书的时候，matlab，python都要放在手边，然后把书里面的知识尽量实践出来。很多教科书里都会在章节后面的习题里放一些写明是用matlab做的习题，要尽量多做一些这样的题。

你可能早就听说了ROS的大名，但是最好不要在大四之前去碰它。因为ROS用了很多操作系统和网络的底层技术。我在知乎回答高手可以谈谈ROS机器人操作平台开发的一些经验吗？ - YY硕的回答里有简单的介绍。ROS的设计目标是把机器人的控制和传感器处理的软件和它的硬件隔离开，用上ROS以后，你可以方便地用到很多能直接跑的软件代码。但是ROS从入门到精通需要至少一年以上的时间，你必须不断地用，不断地尝试新的代码和硬件，才能对它熟悉起来。

ROS的可视化工具Rviz里面对于机器人旋转的表示用的是四元数，而在你之前研究四旋翼飞行器时，里面的代码表示旋转用的是欧拉角，做姿态解算用的可能是四元数。这个时候要开始有意识地去学习旋转表示法之间的区别和联系。

要重视大四期间的实习和毕业设计。很多大四的学生毕业设计都会非常颓地做一下，我觉得是不好的。要把做毕业设计的过程看做一个正式的项目。这个项目除了做好技术方面的工作，也要做好展示方面的工作。中国工程师的一大特点是，不会表达自己，可能做的东西水平很高，但是做出PPT就会犯字体花哨不正式、一页上面字太多，图文没有联系等表达上的问题。通过PPT介绍、展示自己的成果在工程师的职业生涯的任何一个阶段都非常重要，它甚至也一定程度上限制了机器人工程师能够达到的高度。只有能够把自己的成果清晰地表达给自己的团队，才能获得其他人的反馈、通过沟通提高团队的整体凝聚力和知识水平，这样自己在团队能够获得更多的认可，有助于团队整体工作效率的提高。我每年去参加几次学术会议，都在会议上感觉到一些中国的科研人员走到国际上以后，演讲能力很差就会导致他们的科研成果不受重视。当然其他国家的科研人员也是这样。

为了写出美观的技术报告和毕业论文，你可以开始学习Latex。Latex作为国际国内第一写作神器，学习资料在网上有很多。Latex的学习和使用同样也是需要不断地熟能生巧，多写多练就熟悉了。写毕业论文有个问题是怎么做出精美的矢量图，我推荐Draw Freely | Inkscape，一个比Illustrator更轻量化、但是有些功能反而更强大的免费软件。

对于那些想申请出国留学的同学，你要做一个自己个人的成果展示，用网页的形式呈现比较好。把自己Github链接（如果你按我说的，大学第一天就申请Github账号，现在已经是一个三年的老油条了）、做过的机器人视频、写过的技术报告和文章（最好是英文的）放在上面。

关于毕业设计的选题，我推荐这么几个：

1. 手写双目视觉里程计。涉及到图像处理、特征匹配、位置解算、空间变换等等。

2. 手写四旋翼飞行器基于GPS的轨迹规划。涉及到深挖四旋翼飞行器的运动原理、IMU原理、轨迹生成和优化等。

3. 造一个被推了也不会倒的双足舵机机器人。涉及到舵机控制、倒立摆建模、动力学分析、PID控制、IMU原理等。

4. 深度学习训练一个小车追人跑。涉及到深度学习工具包使用、数据集采集、数据集分析、小车控制等。

5. 机械臂给人端茶倒水。这个相对来说土豪一点，因为能直接拿来用的机械臂都很贵，这个要看实验室有没有条件了。涉及到多自由度机械臂原理的学习、工具包的使用、轨迹规划等等。

这几个项目要做好，都要持续投入三个月以上的时间以及一定的资金，每一个都是理论多于实践。当然同学们自己也可以自己选择自己的毕业设计题目，但是最好还是选做出来能跑能飞的东西，同时避免选择需要花大量时间去拧螺丝、焊板子的题目，尽量买现成的电机、开发板、3D打印结构，大四要多给自己留时间去看书和写代码。

大四到研究生之前的暑假，最好去一些比较不错的机器人公司实习一下，比如说大疆。当然你也可以继续做机器人比赛，比如RoboMasters和大疆的飞行器比赛。

研究生一年级

研究生的时候，你的目标比较清晰了，就是做一种机器人至少两年时间，并从中发掘出可以发表论文的知识点。上面我给出的书单里面，你要开始精读里面的几本。

如果做机器人视觉定位、传感器融合方面的研究，1、4、5、7四本书一定要精读。

如果做控制系统的研究，3、8、9、10一定要精读。

如果做规划算法的研究，那么还要读其他偏CS一些的书，比如讲A* search，random forest，图论方面知识的教材。由于我在这方面造诣不深，就不托大了。

如果学习随机系统的控制和最优控制，除了1、2、3，还要读一本神书Stochastic models estimation and control（cs.unc.edu/~welch/kalma...）。

如果研究机器人视觉定位，几种常用的定位算法：PTAM（Parallel Tracking and Mapping for Small AR Workspaces (PTAM)），ROS的标配VO（viso2_ros - ROS Wiki），SVO（GitHub - uzh-rpg/rpg_svo: Semi-direct Visual Odometry），LSD-SLAM（vision.in.tum.de/resear...）ORB-SLAM（GitHub - raulmur/ORB_SLAM2: Real-Time SLAM for Monocular, Stereo and RGB-D Cameras, with Loop Detection and Relocalization Capabilities），都必须自己学习之后全部跑一遍。只会用OpenCV的函数和这些工具包并不能说明你会视觉定位，必须要能自己手写出一个能用的才算。国内有一个很厉害的SLAM专家叫高翔，他的博客要关注一下：机器人 - 标签。由于SLAM这两年很火，研究的人很多，所以网上可以参考的资料也很多，比如github.com/hcdth011/ROS...，就在ROS上实现了几种定位算法的对比。

我现在非常不建议同学们选择从四旋翼飞行器的动力学控制里找问题作为研究课题。因为四旋翼飞行器的特点已经被研究透了。目前国际上对多旋翼飞行器的研究主要集中在造一些奇葩形状的飞行器，以及给多旋翼飞行器上安装一个机械臂去做力控制，这样做就对多旋翼飞行器控制的动力学造成了一些影响。因此需要同学对动力学和多自由度机械臂控制有比较深的认识。

自动导航和驾驶是这两年的热点，一方面汽车的自动化是大势所趋，另一方面多旋翼飞行器异军突起，产生了很多对自动飞行的需求。除了机器人视觉定位算法以外，同学还需要学习其他的传感器，以及这些传感器与视觉定位算法怎么融合。这里面有很多坑，比如计算量的问题，怎么保证融合算法不崩，怎么处理传感器的延时等等，都需要同学结合自己的项目去踩，坑踩得多了才能成长。如果大家想找一个多旋翼飞行器平台研究自动导航，我推荐大疆的M100，我已经在知乎回答RoboMasters2015夏令营是怎样的？ - YY硕的回答里吹过一波M100，前面说过的今年夏令营的知乎回答参加Robomasters 2016夏令营是怎样一种体验? - DJI 大疆创新里也有人帮我吹了一波。

一些大学里学过的知识点，是必须结合研究生期间的项目的需求弄得很清楚的，比如三大变换（傅里叶变换，拉普拉斯变换，Z变换），旋转表示法（欧拉角、四元数、旋转矩阵），数值计算怎么防止矩阵出现数值问题等等。除了自己的项目，还需要把凸优化、卡尔曼滤波还有多自由度机械臂的控制学习一下。这三个领域的知识，是任何一种机器人都会用的到比较难的知识。

凸优化和凸优化的各种变形是非常重要的知识，因为各行各业里的研究问题，多半是会建立一个优化问题去解决的。上面提到的《Convex Optimization》web.stanford.edu/~boyd/...，也是一本神书，同学们一定要认真读一读。Matlab、Python、C++都有一些现成的工具包可以帮助你解优化问题，不过最好同学们能自己手写一些基本的优化算法，比如gradient descend，barrier method等等。另外现在主流的SLAM算法，后端都是通过一种叫做g2o的优化算法来出

效果的。而且g2o能够整合bundle adjustment 和structure-from-motion这两大计算机视觉里的关键问题，可以说是一种很好的计算思想了，非常有必要学习一下g2o。

卡尔曼滤波在上面书单里的1和3都有提到，同时在神书Stochastic models estimation and control（cs.unc.edu/~welch/kalma...）也有相当多的篇幅。卡尔曼滤波有好几种证明的方法，同学最好能自己学会1-2种。

多自由度的机械臂的难点在于机械臂的运动学正反解、运动学控制和动力学控制，基本是一个建模分析和数值算法实现的问题。如果你所在的学校没有一个财力雄厚的机器人实验室的话，你基本上没有机会接触到多自由度的机械臂。这时候之前学到的Simulink和就要学的Gazebo就派上用场了，你可以用Simscape里面的刚体搭一个多自由度机械臂，然后通过Simulink仿真去学习机械臂的控制；也可以用Gazebo的URDF语言写一个机械臂，然后通过Gazebo和ROS的接口去控制机械臂；也可以用ROS里面的著名工具包MoveIt! Motion Planning Framework，不过MoveIt的问题是，他只能仿真运动学，而不能仿真动力学。工业领域对多自由度的机械臂控制通常用一个叫做D-H表示法的建模工具（Denavit），这个东西我并不太会。我只会向同学们推荐我导师的著作《A mathematical introduction to robotic manipulation》（cds.caltech.edu/~murray...）。

有一个非常神奇的事实：《A mathematical introduction to robotic manipulation》这本机械臂控制领域的著名教材的第二章和计算机视觉领域的著名教材《An Invitation to 3-D Vision》的第二章基本是一样的，都在讲旋转表示法。这是因为所有的旋转表示法都可以归纳为一种优雅的李群结构：SO(3)群。而计算机视觉和机械臂控制都涉及到理解刚体的旋转，事实上用计算系统去观测和控制所有的刚体构成的系统，理解旋转都是很关键的问题。旋转表示法应该作为研究生阶段的一个重要学习的知识点。

李群和李代数是刚体旋转表示背后的数学理论，如果想要深挖一些，可以看这篇文章An elementary introduction to groups and representations的前50页（cmls.polytechnique.fr/p...）。这是我自己读着觉得最好的文章，当然网上也有很多其他的介绍。

研究生阶段还要培养的一个能力是借助各种工具仿真机器人系统的能力。显然地，很多机器人系统真的造出来的话造价昂贵，需要在实际制造之前写一个比较真实的仿真系统出来测试算法。我觉得做仿真系统的能力直接衡量了机器人工程师的技术水平。当你开始要搭一个仿真系统的时候，第一步是通过欧拉方程和牛顿方程确定刚体的运动特点，甚至要自己写刚体二阶微分方程；第二步是确定刚体之间的互联关系，设计不同类型的关节，如果有软性连接需要加入弹簧阻尼模型；第三步是确定被仿真的刚体系统会不会和外界产生碰撞或者其他形式的力，如果有的话，需要设计合适的接触力和摩擦力仿真的模型。多旋翼飞行器的仿真是很简单的，不需要考虑什么接触力。但是多自由度机械臂基本都需要仿真接触力，不和物理世界去交互的机械臂只有很小的实用意义。而能够自行运动locomotion系统，比如双足、多足机器人，则涉及到更多的接触力，多

到接触力都会影响仿真系统的数值稳定性。搭建一个仿真系统需要很强的系统建模能力和数值分析的能力，虽然Simulink、Gazebo、Vrep提供了不同程度的工具简化你的工作，但是要让仿真系统能够稳定运行，必须要能深入其中的细节。有些看起来很高大上的仿真工具，比如Nvida的PhysX，在仿真的时候是忽略掉科里奥利力的，如果不理解仿真的本质，可能就会忽略这一个重要的缺陷。

研究生二年级

你的学习计划接近尾声。现在你已经进入了一个很好的状态：看到一个机器人，能够很果断地分析出它用了什么传感器、执行器、计算平台大概是什么量级，他的执行机构能够承受多少力量。看到一个新的算法，能够大约判断清楚它的执行流程，在什么环节做了优化。看到一个新的没学过的知识，能够分析出它和你以前学过的什么知识有联系，你还需要再学什么才能弄明白这个知识点。

研究生二年级要深化第一年学到的那些技术和知识，要做到完整地读过四五本书，五十篇以上的论文。你已经积累了几万行代码的经验，也能熟练地谈论谁家的电机回差小，谁家的电机线性程度好。

你这个时候可以去写作一些论文，也可以开始学习一些更高级的技术和工具，比如用FPGA和GPU优化算法、魔改Linux内核、玩玩液压系统、了解更多机器学习的知识比如强化学习等等。你也可以从计算机图形学或者计算力学里面找到一些帮助你更好进行机器人仿真和系统分析的工具。由于你懂很多机器人方面的知识，你可以给学校的机器人队做指导，或者带队参加一些比较有挑战性的机器人比赛。

写到这里，我就不可能给出很多不同领域的指导了，因为随着学习的进一步深化，我自己熟悉的领域也在收缩。我只能对几个领域给出我的意见。

对视觉定位和传感器融合来说，SLAM急需新的突破，目前通过几何约束去实现loop closure看起来已经走入了死路，没法有更多的发展了，下一步可行的方向是与深度学习进行结合。具体的一些介绍可以阅读行业中大牛的一篇文章computervisionblog.com/...，记述了几个业界大牛们最新的观点。传感器融合技术，目前还有很多问题可以探索，因为传感器的延时、不均匀的信号，会给定位系统造成困扰，如何去除这些干扰，需要建立比较复杂的非线性优化问题，具体可以关注香港科技大学Shaojie Shen的工作。

对于多自由度机械手和机器人的locomotion来说，这里面还有非常多可以探索的研究问题。我前面提过接触力和摩擦力很难仿真，大神告诉我现在没有任何一种工具和理论能把接触力和摩擦力正确仿真出来，因此如何在机器人系统里妥善处理对这些力的控制，就是很难的问题了。现在业界的一个前沿发展方向，也是利用机器学习技术来帮助机器人学会处理这些外力，不过人类目前最优秀的多自由度机器人系统，Berkeley的Brett机器人，叠几块积木就要用十分钟（New ‘dee

p learning’ technique enables robot mastery of skills via trial and error），显然还有很多提升的空间。这方面的问题同学可以关注知乎大神@戴泓楷@周佳骥。

最后我想再强调一遍表达能力的重要性。你可以从自己带的课程和机器人队入手，把自己这几年来学过的知识做成PPT讲给学生们听，然后让他们给你反馈。多做这样的练习，提升自己做演讲的能力，这将来会让你受益匪浅。

研究生二年级之后，你可以准备进机器人公司工作了，也可以根据你自己感兴趣的研究方向申请博士接着努力。由于你已经掌握了广博的知识和技能，你的职业生涯将会大有作为。

结束语

我从2008年展开自己的机器人生涯。那一年波士顿动力刚刚发布他们的大狗机器人，Python还是很小众的语言，Ubuntu 8.04还很不稳定，Chrome还没有多少人知道。在之后的几年中我目击了深度神经网络的复兴，看到波士顿动力的机器人日渐强大，经历了ROS的起源和繁荣，帮助了大疆的崛起，深深为这个产业的未来感到激动。我希望这篇文章能够帮助更多的年轻人进入机器人学的海洋，为未来机器人学的继续发展贡献力量。