doi: 10.3969/j.issn.1006-1576.2011.12.017

机器人水球比赛中的目标跟踪

张君华[1]，石红[2]，王若鹏[2]，谢广明[1]
（1. 北京大学工学院智能控制实验室，北京 100871；2. 北京石油化工学院数理系，北京 102617）

摘要：针对原有机器人水球比赛的视频多目标跟踪技术的不足，提出一种基于 CamShift 算法的带有自适应能力和主动纠错能力的局域目标跟踪方法。通过介绍基本的 CamShift 算法，并引入自动白平衡调节和自适应能力的优化处理对跟踪方法进行改进，对跟踪过程中可能出现的 5 种可能错误情形提出具体纠错方案。实验结果表明：该方法有效且算法处理速度能达到实时图像处理的要求。

关键词：机器人水球比赛；目标跟踪；全局视觉；水下机器人；多机器人协作

中图分类号：TP242.6　　文献标志码：A

Target Tracking in Robot Water Polo Game

Zhang Junhua[1], Shi Hong[2], Wang Ruopeng[2], Xie Guangming[1]
(1. Intelligent Control Laboratory, College of Engineering, Peking University, Beijing 100871, China;
2. Dept. of Mathematics & Physics, Beijing Institute of Petrochemical Technology, Beijing 102617, China)

Abstract: Aiming to the shortage of the existent multiple visional targets tracking technology of the robot water polo game, a local target tracking method is presented based on CamShift algorithm, which is with adaptive ability and active adjustability. Firstly, the basic CamShift algorithm is introduced. Then, the tacking method is improved by including autonomous white balance adjusting and adaptive optimization. Moreover, for the five possible cases which may lead to wrong tracking, correcting mechanism are given in detail. The experiment results show that the obtained method is valid and satisfied for real time image process.

Keywords: robot water polo game; target tracking; global vision; underwater robot; multi-robot cooperation

0　引言

随着人类对海洋资源的探索，水下操作任务越来越多。很多任务具有很高的复杂性，需要有多个机器人来协作完成。多水下机器人协作系统的研究已经成为需尽快解决的科研课题[1-13]。基于此背景，北京大学联合国内多家高校和科研院所共同建立了机器人水球比赛项目[14]。水球比赛除涉及到陆地机器人足球比赛所需的各种技术外，还涉及水动力分析、水下通讯、图像处理、抗干扰技术等多方面的内容，难度更大、技术更复杂。而基于视频的多目标跟踪技术是机器人水球比赛的关键技术之一。

比赛采用的是基于颜色标识的全局目标搜索方法。原有方法具有不能适应环境光照、速度较慢、不能主动纠错等缺点。而 CamShift（continuously adaptive mean shift）是一种典型的运动目标跟踪算法，具有简单易实现，效率极高的优点。因此，笔者提出了一种基于 CamShift 算法的带有自适应能力和主动纠错能力的局域目标跟踪方法，以解决原有方法的不足。

1　硬件体系

水球比赛的场地是一个 3 000 mm×2 000 mm ×350 mm（长×宽×高）的水池，水深 250～300 mm。其中有效比赛范围根据项目不同而略有区别[14]，比赛规则是基于全局视觉的。如图 1，每个场地上方垂直悬挂一个 CCD 摄像头，用来获得实时全局视频信息，并通过数据线传回上位机，上位机对视频信息进行实时处理，根据参赛者事先编写的比赛策略决定机器人的行动，并通过无线通信将指令传送给机器人。

收稿日期：2011-08-15；修回日期：2011-09-23
基金项目：国家自然基金项目（10972003）；机器人技术与系统国家重点实验室开放基金项目（SKLRS-2009-MS-09）
作者简介：张君华（1990—），男，福建人，本科，从事图像处理、模式识别研究。

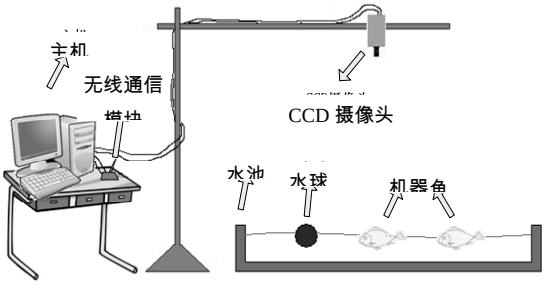

图 1　硬件体系结构图

2　图像处理与目标跟踪

图像处理模块是整个系统中最关键的部分，主要任务就是对 CCD 摄像头采集到的图像进行处理，从中识别出机器鱼和相关任务目标的位姿信息。这些信息是整个系统做出决策与控制的基础，因此图像处理模块的高效性和稳定性十分关键。机器鱼的标记如图 2。

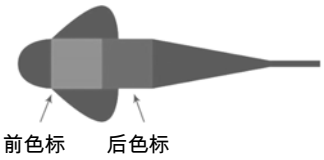

图 2　机器鱼的色标

其中前色标用于区分机器鱼的组别，后色标用于区分同组的机器鱼，2 个色标共同确定机器鱼的位置和方向。CCD 摄像头采集到的一幅典型图像如图 3。

图 3　摄像头采集到的图像

图像处理模块的任务就是要从以上图像中找出各个颜色标记，进而确定机器鱼和球(任务目标)的位置。

目前使用的图像处理方法的大致流程为[4]：

1) 设定一组标准色和每种标准色的替换范围。

2) 将所有颜色位于某一替换范围内的像素替换为相应的标准色。

3) 对每一标准色，统计其在图像中的连续区域的大小和块数，并确定每块的位置。

4) 根据色块的大小判定其是否为鱼的色标，并对距离相近的色块进行配对。如果某对色块的颜色与某条鱼的前、后色标相同，且距离在有效的范围内，则将这对色块识别为这条鱼。

该方法的优点在于，以色块标记机器鱼，特点鲜明，简单而易于识别。但不足之处也很明显：

1) 标准色的替换范围固定，不能很好地适应环境光照不均的问题，机器鱼一旦游到阴影处或高亮处，就可能导致目标丢失。

2) 完全没有利用历史信息，对每帧图像都做全局扫描，因而速度较慢。

3) 如果出现配对错误或目标丢失等问题，只能等待导致错误的因素自行消失，没有主动纠错能力。

3　基于CamShift算法的目标跟踪方法

基于 CamShift 算法的目标跟踪方法选用更接近人眼视觉习惯的 HSV 色彩空间，以像素颜色与标准色的相近程度为灰度，将需要处理的部分图像转化为灰度图，而后使用 CamShift 算法跟踪色块的移动，进而确定机器鱼的运动。一旦发生目标丢失，程序会扫描丢失位置的附近区域，力争找回丢失的目标。此外，随着程序的运行，相应的标准色会不断更新，以适应渐变的环境光照。

3.1　CamShift 算法简介

CamShift 算法的流程如下：

1) 给定一帧灰度图，以及窗口(即算法需要处理的子图像)的位置和大小。

2) 对于窗口内的子图像，以灰度为权重，对每个像素的坐标做加权平均，即计算出子图像的重心。

3) 将第 2) 步的计算结果作为新的窗口位置，返回第 2) 步重新计算，直到 2 次计算得到的重心距离小于给定的阈值。此即被跟踪目标的位置。

4) 将第 3) 步所得的结果作为下一帧图像的窗口初始位置，返回第 1) 步。

因此，笔者先利用 CamShift 算法跟踪每个色标的移动，再根据已知的色标和机器鱼的对应关系确定机器鱼的运动。

3.2　单个色标的跟踪

在本方法中，每个色标拥有 3 个属性，分别为重心位置 $p=(x,y)$、扫描半径 r 和标准色 $c=(h,s,v)$。其中标准色和扫描半径可由该色标在图像中实际呈现的颜色和具体大小而定。由 p 和 r 所确定的圆即为该色标对应的窗口 W。由于 CamShift 算法只涉及窗口内的子图像，故以下所述的所有操作都只需对窗口内的图像进行。

3.2.1 颜色处理

要使用 CamShift 算法跟踪一个色标，首先必须将窗口内的子图像转化为灰度图。图像的处理选在 HSV 色彩空间中进行。将窗口内每个像素的颜色看作随机变量，并假设其服从正态分布

$$N\left(c=(h,s,v),\sigma^2=\left(\sigma_h^2,\sigma_s^2,\sigma_v^2\right)\right)$$

其中 $\left(\sigma_h^2,\sigma_s^2,\sigma_v^2\right)$ 为相应的方差，可事先测量统计得到。由此即可取该正态分布的概率密度作为某一颜色与标准色的相似程度。但由于在灰色(饱和度为 0)处色调不确定，以及在黑色(亮度为 0)处亮度和饱和度都不确定，在这些区域微小的涨落会带来色调的巨大变化，因此在计算中应当排除。

综上，取颜色 $C=(H,S,V)$ 与标准色的相似度函数为：

$$I(C)=f(C,c,\sigma^2)\cdot\eta(S-T_S)\eta(V-T_V)$$

其中：$f(C,c,\sigma^2)$ 为正态分布的概率密度函数；$\eta(x)$ 为阶跃函数；T_S、T_V 分别为饱和度与亮度的阈值。通过函数 $I(C)$，即可将彩色图像转为灰度图像。

3.2.2 色标定位

获得灰度图像后，即可使用 CamShift 算法确定当前色标的位置。但由于机器鱼的移动速度较慢，CamShift 算法的第 2 步实际上只需执行一次即可。

此外，虽然 CamShift 算法常用的窗口为矩形，但在这里却选用了圆形窗口。因为当 2 个同色的色标(比如 2 组的 2 条机器鱼的前色标)距离太近时，矩形的窗口容易导致“漂移”的发生。

如图 4，为了保证完全覆盖一个色标，窗口必须足够大。矩形的窗口空白部分较多，当窗口内挤入另一个同色的色标时，重心就可能逐步偏移，几次迭代后，原本跟踪当前色标的焦点就转移到了另一个色标上。反之，如图 5，圆形的窗口的空白部分较之方形要小得多，因此也就大大降低了“漂移”的可能性。

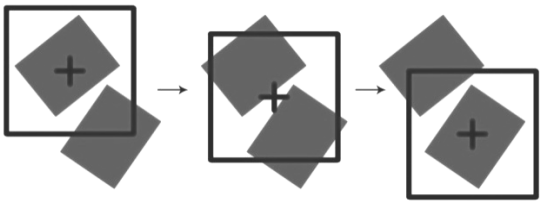

图 4 色标漂移

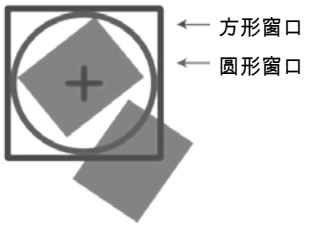

图 5 圆形窗口

3.2.3 白平衡调节

如前所述，本方法以颜色间的相似性作为权值，而色调又是区分不同颜色的重要指标，因此不同的标准色之间的色调差异越大越好，所以色标的颜色被设计为在色相环上均匀分布。但由于环境光源和 CCD 摄像头的设置等原因，图像的白平衡可能不正确，直观表现为图像的整体色调向某一颜色偏移。在这种情况下，不同的标准色之间的差异变小，很容易发生识别错误，因此需要对图像进行适当的白平衡调节。这里采用最方便最常用的线性近似：

$$\begin{bmatrix}R\\G\\B\end{bmatrix}=\begin{bmatrix}P_R&0&0\\0&P_G&0\\0&0&P_B\end{bmatrix}\begin{bmatrix}R'\\G'\\B'\end{bmatrix}+\begin{bmatrix}Q_R\\Q_G\\Q_B\end{bmatrix}$$

其中：(R',G',B') 为调节前某个像素的颜色；(R,G,B) 为调节后对应像素的颜色。参数 P、Q 可以通过简单尝试确定，在 CCD 摄像头和环境光源不变的情况下不变。

图 6 是一帧图像调节前后的对比。

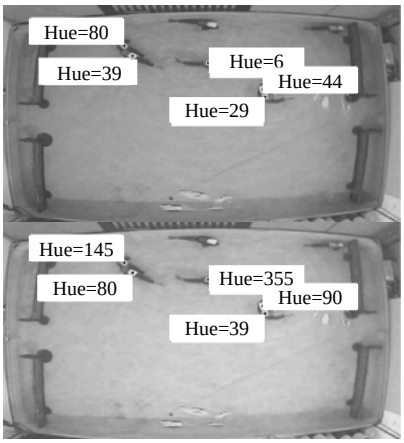

图 6 白平衡调节

图 6 中，调节前，各色标的色调明显集中在橙色(Hue = 40)附近，而调节后则分布得更加均匀。

3.2.4 自适应功能

由于环境光照的不均匀和鱼在游动、碰撞时出现的倾斜，色标呈现的颜色(尤其是亮度)会有变化，因此色标的标准色必须不断修正以适应变化。笔者

采用对窗口内的颜色加权平均的方法进行解决。

$$c=\frac{k\cdot c'+\sum_{(x,y)\in W} I(C(x,y))\cdot C(x,y)}{k+\sum_{(x,y)\in W} I(C(x,y))}$$

其中：c'为每次修正前的标准色；k为阻尼系数。加入阻尼系数可以防止标准色在某些情况下发生突变(例如色标因机器鱼翻身而消失)。

3.3　机器鱼的跟踪及错误纠正

上面讲述了如何对 1 个色标进行跟踪。要跟踪 1 条机器鱼，只需如图 7 所示跟踪属于它的 2 个色标即可。

图 7　跟踪机器鱼

然而，事实上存在许多特殊情况，会使得跟踪出现错误，例如机器鱼翻身、色标发生镜面反射和多条机器鱼扎堆等等。一个色标跟踪错误可以分为以下 2 种：

1) 色标丢失：丢失跟踪目标，表现为窗口内的灰度值总和过低。

2) 色标漂移：跟踪焦点漂移到其他物体上，表现为同一条鱼的 2 个色标间距不正常。色标漂移一般不能在第一时间被发现。

对于一条机器鱼的 2 个色标，可能存在“丢失，正常”、“漂移，正常”、“丢失，丢失”、“丢失，漂移”和“漂移，漂移”5 种错误情况。下面分别讨论每种情况的判定条件和修正策略。

3.3.1　“丢失，正常”情形

这种错误情况表现为其中一个色标的总灰度值过低，另一个正常。

此时有一个色标的位置是可靠的，因此可将其窗口外围$r\le R\le 3r$的圆环作为已丢失色块的窗口，按跟踪单个色标的方法定位已丢失的色标，如图 8。

图 8　找回已丢失的色标

3.3.2　“漂移，正常”情形

这种情况下 2 个色标的总灰度值都正常，但其间距不正常。

由于程序无法判断到底是哪个色标发生了漂移，可以分别假设其中一个是正常的，在其周围按“丢失，正常”的纠正方案寻找另一个，最后取 2 组结果中总灰度值较大者为最终结果。

3.3.3　“丢失，丢失”情形

这种情况的表现为 2 个色标的总灰度值都过低，一般会在机器鱼翻身或者光照条件突变时发生。一般来说，导致这种错误的原因不会马上消失，而当色标恢复到可被识别的状态时，机器鱼又发生了位移，因此丢失前的色标位置并不可靠。要纠正这一错误，就必须在一个较大的范围内进行扫描，同时搜索前、后色标的位置。

解决此问题的基本思路就是不断枚举窗口位置并进行定位。但对于一个$m\times n$的矩形扫描范围和一个$a\times b$的矩形窗口，这么做的时间复杂度为$O(mnab)$，显然效率极低。这里给出一个加入预处理过程的改进算法：

1) 以机器鱼丢失前的位置为中心，选取一个矩形区域为扫描范围。

2) 将给定的扫描范围内的图像分别按前、后色标的标准色转为灰度图G_1、G_2。

3) 灰度值预处理：

$$\begin{aligned}&U_i(x,y)=U_i(x-1,y)+U_i(x,y-1)-\\&\quad U_i(x-1,y-1)+G_i(x,y);\\&U_i(0,0)=U_i(0,y)=U_i(x,0)=0;\\&(1\le x\le m;1\le y\le n;\quad i=1,2)\end{aligned}$$

4) 坐标预处理：

$$\begin{aligned}&V_i(x,y)=V_i(x-1,y)+V_i(x,y-1)-\\&\quad V_i(x-1,y-1)+(x,y)\cdot G_i(x,y);\\&V_i(0,0)=V_i(0,y)=V_i(x,0)=(0,0);\\&(1\le x\le m;1\le y\le n;\quad i=1,2)\end{aligned}$$

5) 枚举扫描范围内的像素(x,y)，分别取该像素周围边长$6r$、$2r$的方形为窗口，计算总灰度值：

$$\begin{aligned}&W_1(x,y)=\\&U_1(x+3r,y+3r)-U_1(x-3r,y+3r)-\\&U_1(x+3r,y-3r)+U_1(x-3r,y-3r)\end{aligned}$$

$$W_2(x,y)=$$
$$U_2(x+r,y+r)-U_2(x-r,y+r)-$$
$$U_2(x+r,y-r)+U_2(x-r,y-r)$$

6) 记录使 $\min(W_1(x,y),W_2(x,y))$ 取最大值的像素坐标 (x^*,y^*)。则前、后色标位置为：

$$p_1=[V_1(x^*+3r,y^*+3r)-V_1(x^*-3r,y^*+3r)-V_1(x^*+3r,y^*-3r)+V_1(x^*-3r,y^*-3r)]\cdot\frac{1}{W_1(x^*,y^*)}$$

$$p_2=[V_2(x^*+r,y^*+r)-V_2(x^*-r,y^*+r)-V_2(x^*+r,y^*-r)+V_2(x^*-r,y^*-r)]\cdot\frac{1}{W_2(x^*,y^*)}$$

以上算法利用适当的预处理，省去了对窗口内像素的求和过程，易知其时间复杂度为 $O(mn)$。

3.3.4 “丢失，漂移”情形

这种情况比较少见，仅当一个色标发生漂移而未被发现，同时另一个色标又因故丢失时才会出现。其表现与“丢失，正常”相同，区别仅在于多次定位所得的总灰度值均过低，即无法在“正常”色标附近找到丢失的色标。

由于这种错误发生条件的特殊性，发生漂移的色块的焦点一定还在正确位置附近，因此可以视同“丢失，丢失”情况进行纠正。

3.3.5 “漂移，漂移”情形

这是最少出现的错误情况，其表现与“漂移，正常”相同，但在 2 个色块附近都找不到总灰度值正常的结果。此时色标的位置信息已经完全不可靠，应当按照“丢失，丢失”的情况处理。

4 实验结果

4.1 正常识别的情形

如图 9，黑色十字表示前色标的位置，白色十字表示后色标的位置，周围的圆形是定位的窗口，窗口中的图即是颜色处理步骤中给出的灰度图。

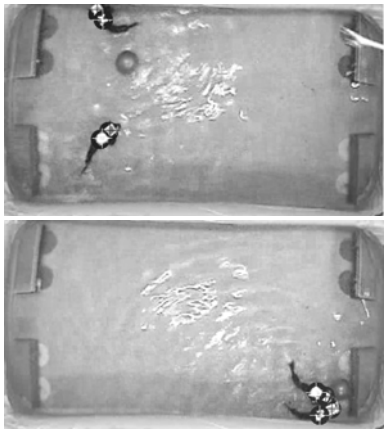

图 9 正常识别

4.2 错误纠正的情形

图 10 中的黑色环形区域即为搜索时使用的环形窗口，白色大十字表示找回的后色标位置。

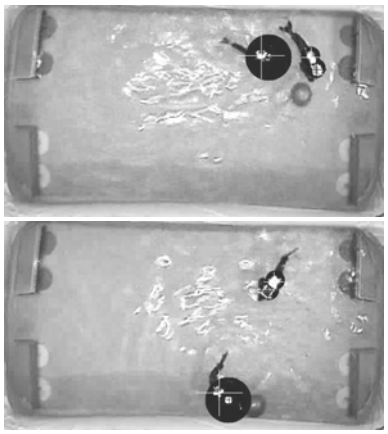

图 10 错误纠正

图 11 是程序处理图像的帧率曲线，可见其计算速度基本在 27 帧/s 左右，只有遇到错误情形时才有所波动。这一速度要高于常用的图像采集设备的采集速度(25 帧/s)，因而该算法可用于实时处理。

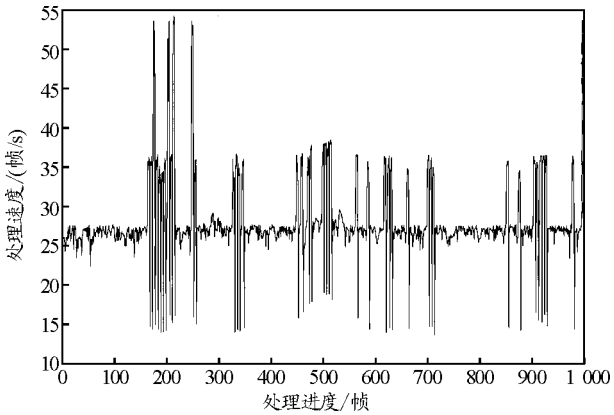

图 11 帧率曲线

5　结论

基于 CamShift 算法的局域目标跟踪方法解决了原有方法的不足，具有自适应能力和主动纠错能力，并通过实验验证了该目标跟踪方法的有效性。

参考文献：

[1] Hu Yonghui, Zhao Wei, Xie Guangming, et al. Development and target following of vision-based autonomous robotic fish[J]. Robotica, 2009, 27(10): 1075−1089.

[2] Fan Ruifeng, Yu Junzhi, Wang Long, et al. Optimized design and implementation of biomimetic robotic dolphin[C]. USA: IEEE International Conference on Robotics and Biomimetics, 2005: 484−489.

[3] Shao Jinyan, Wang Long. Plantform cooperation of multiple robotic fish – Robofish Water Polo[C]. UAS: IEEE Conference on Decision and Control, 2007: 1423−1428.

（下转第 78 页）