JULIUS ELINSON MAY 5, 2013

IMAGE-BASED ROCK-CLIMBING SIMULATOR

Background

Rock-climbing is a sport that presents both a physical and athletic challenge. At indoor gyms, climbing walls contain a number routes with varying difficultly that are delimited often by the color of the grips. The difficulty is determined by the number of grips, their sizes and positions, as well as contours in the wall itself. To complete a route, one must use only the designated grips with their hands and feet to ascend to the top; the goal is simply to get to the last grip and thus a climber can use any subset of the available grips and in any order. Thus a route is a set of grips, whereas a path is a ordered list of tuples of grips, where each tuple represents the positions of each limb at any given position. Climbing requires both the physical stamina to ascend the wall, but also constructing a viable path where one's body weight is balanced and each move can be completed given the physical constraints of each climber.

Objectives

The goal of this project was to simulate a rock climber by generating a physically-feasible path to solve a route based on an input image. The program is named jug, after a type of hand-grip that is very easy to grab and apply weight to. As part of that objective, the program had to be able to identify a particular route within an image, analyze each grip in the route, model basic human motion and perform an intelligent search to find a path to the top. To limit the scope and difficulty of the project, only routes without overhang[1] were considered, so that motion and forces would be limited to the 2-D plane.

In order to simulate a human interacting with grips, the program had to be able to analyze a wide array of grip types. Different grips at different orientations can support different forces applied on them. Moreover, variations in size, texture and shape have implications for what and how many limbs can be put on a particular grip at a given time. Additionally, analyzing a climber's position needed to consider each limb in the context of the others in order to respect physical constraints of the human body, as well as analyze forces and weight distribution in the aggregate.

The search algorithm had to account for the physical implications of a possible path. While a shortest path offers certain advantages, the number of moves also had to weighed against the physical strain induced by each move.

[1]A overhang is when a wall juts out so that climbing it would cause the climber to become supine.

Implementation

Tools

Jug was developed in C++ using both Qt & OpenCV. Qt provides a number of convenience classes for streamlined developed and OpenCV is an excellent library for image processing and provides some basic graphical user interface support. The program is cross-platform and uses CMake for the build process.

Image Processing

Jug comes with several example climbing wall images and it can load user-specified files. The program starts by prompting the user to select a grip with their mouse that is contained in the route they wish to solve. Using this, the RGB image is first converted to HSV space. The reason for this is that hue is generally more invariant between shades of the same color then in the entire RGB space, and grips are often different color shades or have whiter tints due from chalk residue. The hue value of the pixel selection by the user is then used to threshold the entire image using a small tolerance range.

The result of thresholding is a binary image that is noisy, as there are often holes in regions or small speckles of white pixels that do not correspond to interesting regions. To correct this, an erosion and dilation algorithm is used. In this process, the perimeter of any region is eroded several pixels and then expanded for twice as many pixels, and finally eroded once again. For small speckles, the first erosion process will remove the region entirely so that it will disappear and hence not be dilated. For holes within regions, the dilation process will cause the hole to fill and then there will be no perimeter there when the erosion occurs again. For all the other regions, the net effect of the three steps is no change. The contour of each remaining region in the binary image is used to represent a grip.

Grip Analysis

Using the grip contour, a number of intrinsic properties of the grip are determined. The area, perimeter, and center of mass are all computed, as well as several more complex representations. One such calculation is of the grip's convexity defects, which is performed by analyzing the gaps between the contour and its convex hull. This yields the number of such defects as well as their sizes and relative locations.

Additionally, to analyze the grip's orientation, the program computes what I refer to as the normal field. This is a tally of the normal vectors along the contour. Because the contour is a contiguous, 8-connected[2] discrete set of pixels, the normal vectors are computed on each pixel by computing the line orthogonal to the vector connecting the previous and next pixel and thus there are only a small set of possible orientations. The normal field captures the relative sizes of the grip sides in each direction.

[2]Pixels are 8-connected if all 8 surrounding pixels, including diagonals, are considered neighbors.

Physical Engine

In jug, a human is modeled as 4 limbs and a center of mass. Half the climber's weight is in the center and the other half is distributed evenly over each limb. The forces are computed at the center of mass such that an arm extending diagonally out and down will contribute both a horizontal and vertical force.

The feasibility of a position, which is a ordered list of 4 grips representing the locations of the limbs, is determined by both analyzing the forces, as well as simpler binary criteria. These criteria sanity check the climber's position and filter out anything that is unrealistic. The checks for this include whether each limb is both within a certain distance from the center of mass, as well as not too close to it.