Stalker’s Blade

David Matz

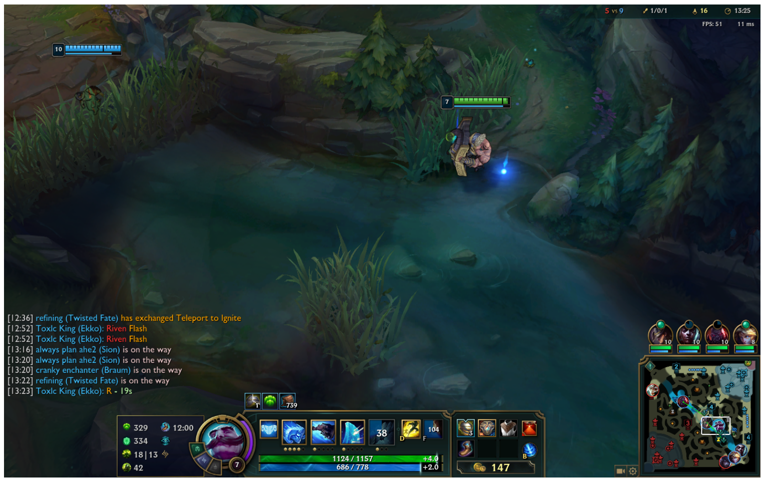
1 Motivation

In *League of Legends*, Stalker’s Blade is an item used by junglers. Unlike other roles which usually go to the same lane during the early game, the jungler has the freedom to play PvP or PvE in many different places on the map. As such, players care strongly about jungler decision making, known as “pathing”, because where the jungler goes has an impact on them and their teammates.

Since where the jungler goes on the map is often the difference between a win and a loss, this data is not available from the Riot API to prevent sites from conferring a competitive advantage. So it would be extremely useful if, instead of getting the information from the API, a bot could watch the replay of a game as a human would and follow the jungler, then efficiently visualize that information for players.

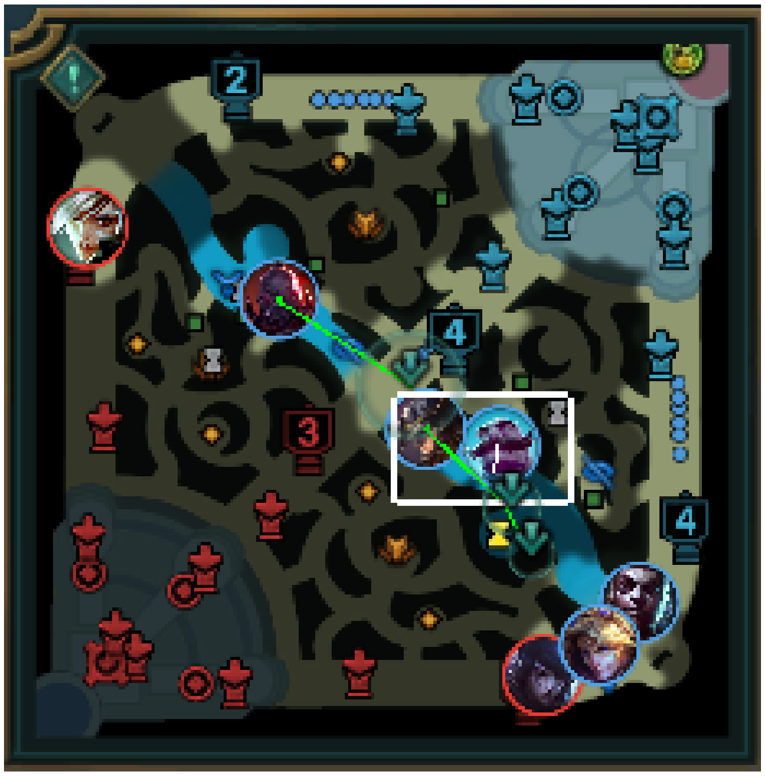
2 Design

This is what the screen looks like in a game of *League*, but what we’re interested in is the minimap in the bottom right corner, which shows player position:



```
[12:36] refining (Twisted Fate) has exchanged Teleport to Ignite
[12:52] Toxlc King (Ekko): Riven Flash
[12:52] Toxlc King (Ekko): Riven Flash
[13:16] always plan ahe2 (Sion) is on the way
[13:20] always plan ahe2 (Sion) is on the way
[13:20] cranky enchanter (Braum) is on the way
[13:22] refining (Twisted Fate) is on the way
[13:23] Toxlc King (Ekko): R - 19s
```

A close up of just the minimap:



The circles with faces are champion portraits, and it's their position we want to get. Online, people have proposed using machine learning to classify a champion icon and position its bounding box, which might be good but is sort of overkill imo. Being either lazy or clever, I'd note that the camera already draws a white rectangle on the minimap, so if you just lock the camera on a champion then the rectangle will follow them around the minimap. So the plan is to follow the camera box around.

3 Implementation

You can read the code here, but it accounts for various challenges that aren't self explanatory, so it's worth reading how it works. It takes a file and if they were on blue or red side, such as:

```
python3 stalk.py karthus.webm red
python3 stalk.py jarvan.webm blue
```

The *League* client records clips from game replays as webm. Examples here and here.

It helps to know which side the player is on because when players recall, respawn, or initially spawn, whether they're red or blue side determines which corner of the map they'll appear in. It also determines where their health and mana bars are, which are important stats for junglers early game but aren't available from the Riot API, so this tracks the percent of health and mana bars filled with green and blue pixels respectively.

The main goal is to find the most likely location of the white rectangle representing the camera box around the player. To reduce the space on the map necessary to search, note that the jungler will never appear far from their current location unless they die or recall, so the two places that need to be looked at are around their previous location and in their base.

Unfortunately, when a player dies the replay camera starts following other players around. This is another reason it's important to track their health bar, because when it empties we need to stop recording player position until they respawn.

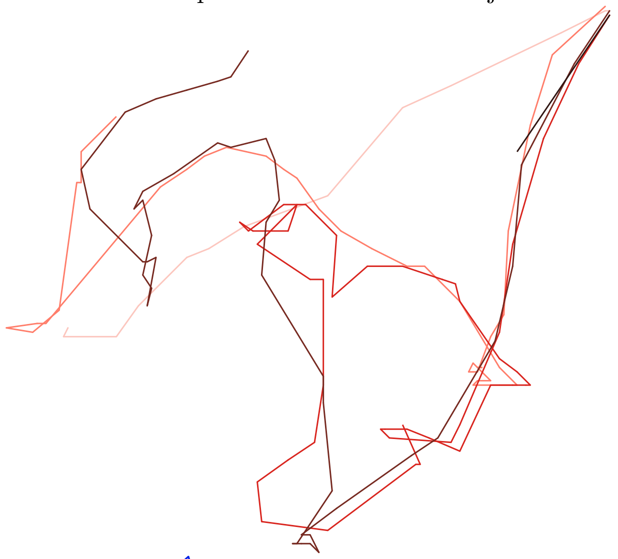
Having tracked the jungler's position, the remaining (probably equally significant) challenge is to visualize it in a way players can quickly read.

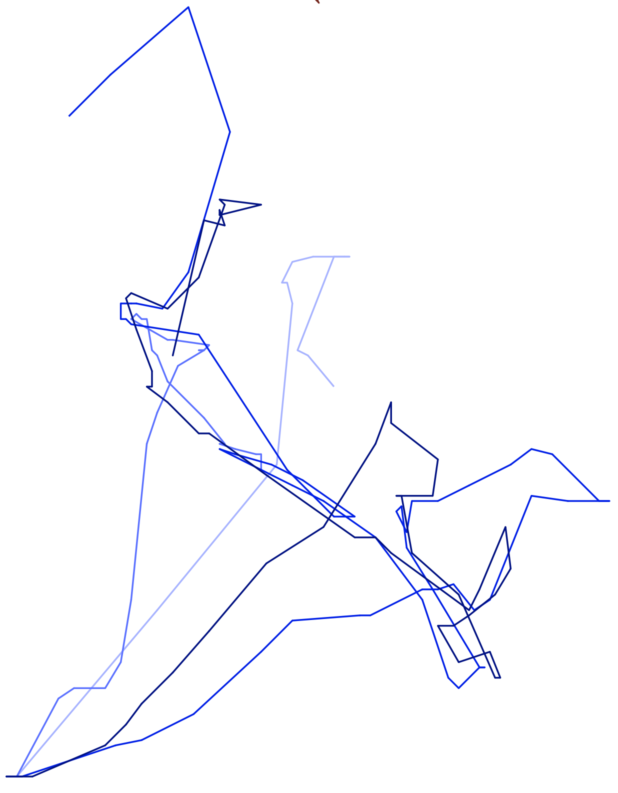
Since this is about movement around the map, the obvious choice is to simply represent x and y coordinates on the minimap by . . . overlaying them on the minimap. Hugely creative. It is very legible for players used to looking at the map, though. The two problems with this approach are it getting messy and trouble distinguishing time as well as place.

Drawing lines from one position to the next as an svg makes it cleaner, but it's still messy. One issue is giant lines straight to base when they recall or die, so instead of drawing those it starts a new line from base. But it's still very squiggly and hard to look at, so I make the lines a bit straighter by replacing lines going to every point with a single straight line which is close-ish to each point it replaces. This is a bit arbitrary but makes the image much more legible.

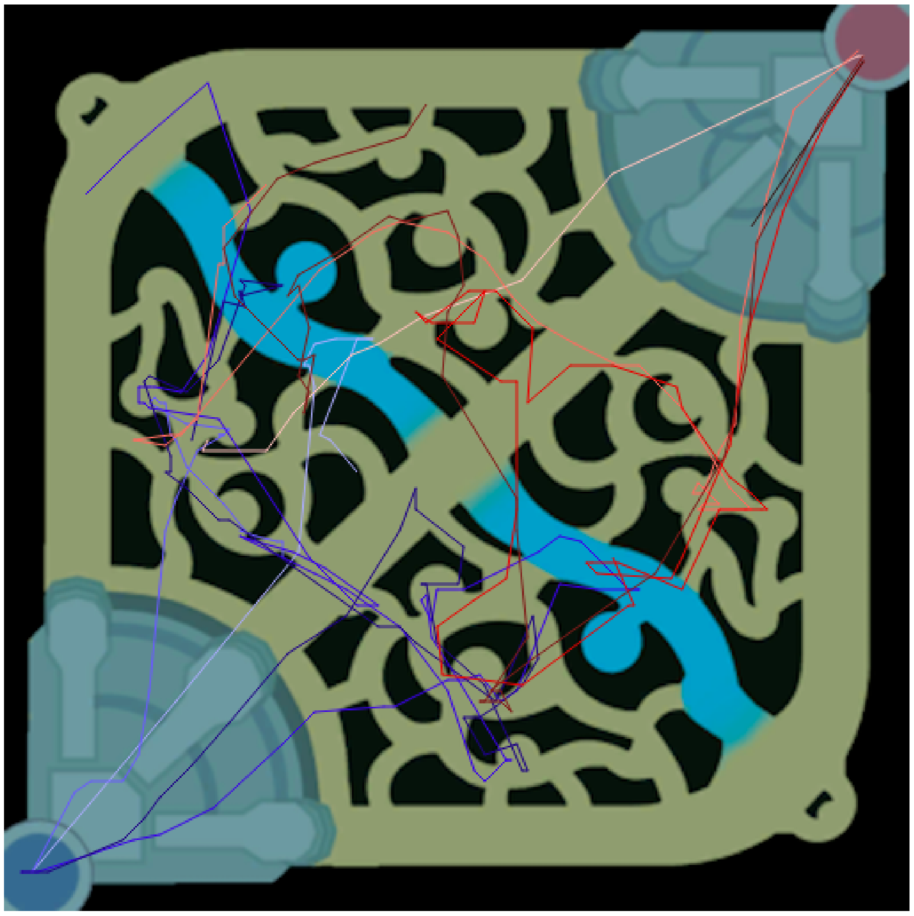
In showing position over time, X and Y coordinates naturally map to X and Y coordinates, so I needed another dimension to map time to. I chose to use a color gradient, moving from white to black through either red or blue depending on the side. I first tried making the color the exact result of a linear interpolation from time to color. However, it's usually possible to follow an uninterrupted line around, whereas it can be harder to tell when a jungler teleports back to base which path they take next, so I ended up using an abrupt jump in color every time the jungler returns to base.

Here are the outputs for the karthus and jarvan examples:





And together, overlaid on the minimap:



It's still not beautiful, but it's somewhat intelligible! Yay! The story of this game is:

Light pink/blue lines: both junglers fight in top river lvl 1, red team wins and karthus goes to enemy blue buff.

Pink line: karthus goes to blue and clears bot to top, then invades enemy topside

Red line: karthus goes to bot scuttle, clears gromp wolves raptors while hovering mid, then invades enemy red

Brown line: karthus again goes botside, mid, and to top river/scuttle

Jarvan, meanwhile, was mainly clearing own his own jungle, sometimes walking into the river and after his second clear ganking top.

4 Future Work

It would be easier to figure out what happened and who won early by including kill events. These are available from the API and can be seen on sites such as op.gg (i.e. player(s) killed other player at this postition and time). That way you know not only did the jungler make this decision, it was a success (or failure).

It would also be nice to provide health, mana, and lane proximity stats for the early game, which I have computed from tracking the jungler and are not available from the API.

Setting up a system to record jungler paths automatically would entail a machine that can output to a screen so that the *League* replay system can work. The cost of doing this seems prohibitive, which is probably why no stat sites such as op.gg currently offer pathing. But finding a way to make this publicly available for, say, a limited selection of high elo accounts, would probably change how these players analyze their games.