Extracting and Composing a Dataset of Competitive Counter-Strike Global Offensive Matches

Erick Rocha[1], **Henrique Maio**[2], **Daniel S. Menasché**[1], **Claudio Miceli**[2]

[1]Instituto de Computação (UFRJ) RJ – Brazil

[2]Escola de Engenharia (UFRJ) RJ – Brazil

erickkrocha@gmail.com, henriquemaio@poli.ufrj.br,

sadoc@ic.ufrj.br, miceli@nce.ufrj.br

Abstract. *There is a growing necessity for insightful and meaningful analytics within eSports: be it to entertain spectators as they watch their favorite teams compete, to automatically identify and catch cheaters or even to gain a competitive edge over an opponent, there is a plethora of potential applications for analytics within the scene. It follows, then, that there is also a necessity for well-structured and organized datasets that enable efficient data exploration and serve as a foundation for visualization and analytic layers. Our work provides the means by which to construct such a dataset for the Counter-Strike Global Offensive (CS:GO) game.*

1. Introduction

From the ages of ancient civilizations to the current times, the practice of sports has always been a widespread activity and a source of recreation for the humankind [El-Harami 2015].

With the development and evolution of technology, the same atmosphere that has ignited crowds of spectators and inspired people all over the world in sports, has also become a part of electronic games. By breaking the frontiers and allowing people to not only spectate, but also compete with other people around the world, competitions that until around the 2000s were mostly among amateur players, started to attract the market's attention. Rapidly following it, a large structure was formed around professional competitions, which became collectively known as "eSports".

According to SuperData's annual report [SuperData 2021], digital games earned $126.6B in 2020 and many organizations have been investing in the creation of teams and leagues to explore that potential. One of the games that have for a long time been a pillar of the eSports world and that greatly influenced its growth is the franchise of the multiplayer first-person shooter: Counter-Strike (CS).

Released in 1999 as a modification[1] of another game called Half-Life, the Counter-Strike franchise has surpassed and outlived its predecessor, with many versions of the game being developed and released. Its latest version - called "Counter-Strike: Global Offensive" (CS:GO) - is the subject of this article. To break the game (in its competitive scene) down in simple terms:

[1]Modifications are also known as mods in the gaming jargon.

- Each match is disputed by two teams composed of five players.
- Each match is usually composed of multiple maps that are disputed in a best-of-X format (with X being the number of maps played, usually ranging from 1 to 5).
- Each map is won by the first team to reach 16 rounds.
- After playing 15 rounds, the two teams switch sides (T-side to CT-side and vice versa).
- Each round is 1 minute and 55 seconds long, with both teams having specific win conditions:
  - The T-side can win a round by either eliminating every member of the opposing team or by planting a bomb in a designated zone, which will detonate after 40 seconds.
  - The CT-side can win a round by either eliminating every member of the opposing team or by defusing the bomb after it has been planted.

On the tail of the incredible growth of the eSports market and of the CS:GO competitive scenario is the ever-growing need for insightful analytic and visualizations. Acquiring data to support these goals is not easy: the readily available data one can find in most CS:GO websites can only support basic analysis. Furthermore, there are many players[2][3] in the market that seemingly employ sophisticated data retrieval strategies to obtain quality data, but the knowledge and tooling used to obtain such data seems to be intellectual property. This ends up leaving a gap with respect to publicly available datasets.

Thus, to enable data exploration and ultimately empower the community to conduct meaningful and insightful analysis, this article introduces a workflow that can be leveraged to retrieve quality CS:GO data.

Outline. The rest of this article is structured as follows. Section 2 presents the challenges, Section 3 describes the sources of data available and Section 4 proposes a workflow that can be executed to build the dataset. The dataset itself is described in Sections 5 and 6. A simple data analysis is described in Section 7 to further motivate the reader on what can be achieved by leveraging the dataset. Related work is presented in Section 8, dataset availability in Section 9 and Section 10 concludes.

2. Challenges

Collecting data in a large scale fashion can be a problem even if the data is readily available: there are many avenues of concern within the overall process of collecting, storing and shaping raw data into valuable information. For instance, how do you optimize and scale your data fetching routine when the source isn't well-structured or protects itself against crawlers? How do you store the data while minimizing infrastructure costs and maximizing retrieval response time? Will the data require real time aggregation? Should it be structured with a relational schema or not? There are many more questions looming over the topic, and answering these questions requires a deeper understanding of one's objectives when handling the data.

The collection process for CS:GO data is no different. There are many challenges in capturing and structuring the data in a useful way. To name a few:

[2]`https://sixteenzero.net/`
[3]`https://csgostats.gg/`

- Surpassing anti-crawling mechanisms.
- Scaling the data fetching routine.
- Extracting information out of CS:GO “demo” files.
- Linking low and high level data (see Sections 4.5 and 5).
- Managing associated costs.

The breadth and depth of data available in a CS:GO match is astounding, though. Once the challenges are overturned, it is possible to build a robust dataset that can be leveraged in many ways, such as to gain insight into aspects of the game that are hidden from the “naked eye”, to define new models that better evaluate the impact of each player in a match and to find trends in the game.[4]

3. Data Sources

To build a truly meaningful CS:GO dataset, we believe one has to leverage aspects from two different data sources and bridge the gap in between them:

- The first source – containing data henceforth referred to as “high-level data” – is any website or application that tracks competitive matches and displays their results. It offers information such as the date window in which championships took place, the dates in which matches were played, the teams and players that participated in those matches, their scores and finally, some high-level statistics that can support shallow analysis.
- The second source – containing data henceforth referred to as “low-level data” – is an actual “replay” file of a match. These “replay” files – henceforth referred to as “demo” files, as they are known in the community – are protobuf-serialized files that contain every important event that occurred within the match. It offers a plethora of information regarding every player at almost any given moment in the game, such as their positions, their net worth (i.e. current money plus investment in current equipment) and their actions (e.g. movements, shots, grenades thrown).

Thus, for the overall success of our endeavor, it was paramount to select a source that made it possible to capture both high and low-level data. Consider the HLTV CS:GO portal[5]: in it, one can not only browse championships and select the results of the matches played in them, but also download the “demo files” of each match.

4. Workflow

Next, we propose a workflow that can be leveraged to compose a CS:GO dataset capable of bridging the gap between both high and low-level data planes. It consists of five different components:

4.1. Crawler

The web crawler is the entry point of the workflow: given the identification number of a championship, it crawls the HLTV website (1), scrapes relevant data and stores it in a manifest file (2). The information stored in this file is imperative for the construction of the dataset and is used in different steps of the workflow. It stores high-level information

[4]Such trends also known as “meta” within the gaming community.

[5]`http://hltv.org`

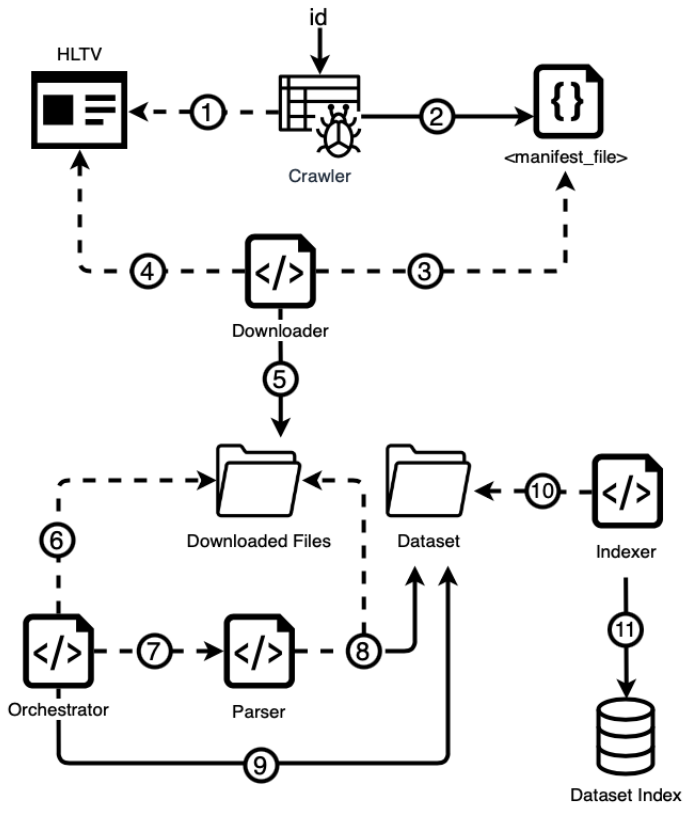

Figure 1. Dataset generation workflow

such as the teams and players participating in the championship, the results of each map disputed and finally the download URL for each "demo file" - the source of low-level data.

4.2. Downloader

The downloader is the module responsible for downloading the "demo files" from HLTV. It retrieves the download URLs from the manifest file (3) and downloads (4) each file to the configured directory in the local file system (5).

4.3. Parser

The demo file parser parses the downloaded protobuf-serialized "demo files" and writes the desired data in a set of CSV files (8).

4.4. Orchestrator

The orchestrator is the module responsible for orchestrating the transformation of the "demo files" into the final parquet files, which contain all the low-level data we deem important for analysis. It does this by iterating over the downloaded "demo files" (6), invoking the parser (7) and finally converting the data from CSV to the parquet columnar format (9).

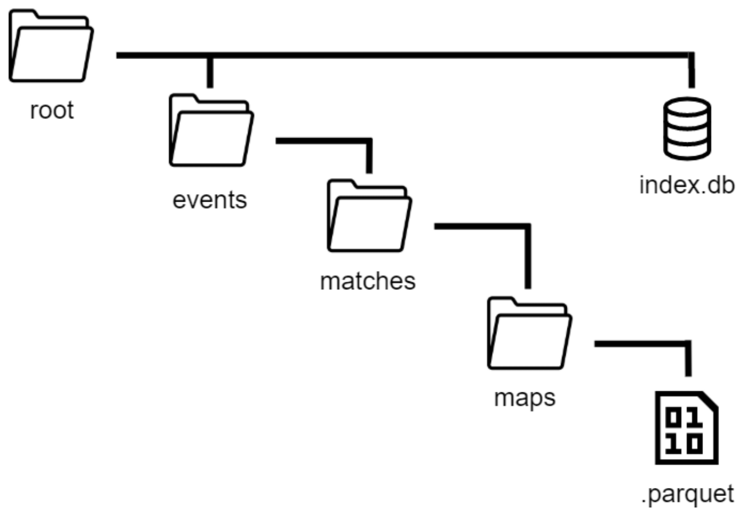

Figure 2. Dataset Hierarchical Structure

4.5. Indexer

The indexer is the last module of the workflow: it is responsible for bridging the gap between the high and low-level data planes. It does this by indexing the location of each disputed map data - already structured as a set of parquet files - in the local file system ⑩, wrapping it around high-level data read from the manifest file and finally exposing this information through a "queryable" interface: a SQLite database ⑪.

5. Dataset Structure

The dataset was designed to be consumed locally while minimizing the resources required to store and process it. The main drivers behind the design were to allow cost-effective data exploration and to facilitate integration with well established data-science libraries (e.g. Pandas). It is structured as a set of files on the local file system and is composed of a SQLite database and a set of parquet files.

5.1. SQLite File

As outlined in the previous section, the SQLite database is the link between high and low-level entities: through it one can perform queries using high-level information - such as a player or a map name - and retrieve the location of relevant parquet files containing the low-level in-game data that enables in-depth analysis.

5.2. Parquet Files

The parquet files contain low-level data, such as the events that occurred within the game and the positions of each player at virtually every moment of the game. As Figure 2 shows, these files are laid out in a directory hierarchy consisting of:

1. A directory for each event (championship);
2. A directory for each match played within each event;
3. A directory for each map played within each match;
4. The parquet files.

6. Dataset

Next, we describe the obtained dataset and dive into the contents of what we have been referring to as low-level data. We restrict our description, focusing on a proof-of-concept of what can be achieved by leveraging the workflow described in Section 4. Lastly, we cover the resources required to make use of this proof-of-concept dataset.

Our dataset contains all matches played in the "ESL Pro League Season 13" [6] championship, where 73 distinct matches were disputed throughout the competition, resulting in a total of 173 maps played. Each disputed map has its in-game information partitioned in a set of five different parquet files, further described in the next subsections.

6.1. bomb_lifecycle.parquet

A "bomb lifecycle" file contains events related to the C4 explosive - a mechanism in the game that can ultimately be used to claim a round. If the bomb is defused or isn't planted at all in a round's allotted time, the counter-terrorists win. If the C4 explosive explodes, the terrorists win. Table 1 shows a sample of the file contents.

Table 1. Bomb lifecycle

tick	round	event	userId
6877	0	bomb_dropped	10
7260	0	bomb_pickup	23
16903	0	bomb_planted	23
22152	1	bomb_exploded	23
22792	1	bomb_pickup	21

6.2. player_death.parquet

A "player death" file contains the events of every death that occurred throughout a particular game, along with some information to contextualize each death, such as whether the player was blinded by a flash-grenade, which weapon was used, whether the bullet was a "headshot", etc. Table 2 shows a sample of the file contents.

Table 2. Player death

tick	round	userId	attacker	assister	assistedFlash	weapon	headshot	penetrated
5497	0	13	10	23	False	glock	False	0
6216	0	21	18	0	False	usp_silencer	True	0
6877	0	10	19	0	False	usp_silencer	True	0
7087	0	19	23	0	False	glock	True	0
7503	0	9	18	0	False	usp_silencer	True	0

6.3. tick.parquet

A "tick" file contains basic player information at virtually every moment of the game, such as their position on the map and their current health points[7]. Table 3 shows a sample of the file contents.

[6] https://www.hltv.org/events/5553/esl-pro-league-season-13

[7] Each player begins each round with 100 health points and their death means those points were reduced to 0

Table 3. Ticks

tick	round	userId	steamId	userName	health	pitch	yaw	speed	x	y	z	placeName
4350	0	9	xxx	blameF	100	4.48	106.81	0.0	1296.0	-256.0	-167.96	TSpawn
4350	0	10	xxx	jks	100	11.14	108.93	0.0	1216.0	-16.0	-163.96	TSpawn
4350	0	11	xxx	RUSH	100	11.66	317.64	0.0	1216.0	-211.0	-163.96	TSpawn
4350	0	12	xxx	tiziaN	100	9.87	87.79	0.0	-1656.0	-1800.0	-266.92	CTSpawn
4350	0	13	xxx	tabseN	100	6.55	89.04	0.0	-1552.0	-1808.0	-266.29	CTSpawn

6.4. utility_lifecycle.parquet

A "utility lifecycle" file contains information regarding the utility grenades used throughout the match. The coordinates in this file represent where the grenade landed and exploded or expired. Table 4 shows a sample of the file contents.

Table 4. Utility lifecycle

tick	round	event	userId	x	y	z
4622	0	smokegrenade_thrown	11.0	1422.96	-367.96	-165.65
5307	0	flashbang_thrown	11.0	1180.12	-964.68	-261.65
5529	0	flashbang_detonate	11.0	-88.27	-1686.01	200.85
5842	0	smokegrenade_detonate	11.0	-640.25	-1582.35	-165.96
8153	0	smokegrenade_expired	11.0	-640.25	-1582.35	-165.96

6.5. weapon_fire.parquet

A "weapon fire" file contains context information regarding every weapon that was fired throughout the match: the actions of throwing a grenade, firing a weapon or using a knife are all classified as a "weapon fired" event. Table 5 shows a sample of the file contents.

Table 5. Weapon fires dataset

tick	round	userId	weapon
4622	0	11	weapon_smokegrenade
4722	0	12	weapon_knife_survival_bowie
4796	0	12	weapon_knife_survival_bowie
4879	0	13	weapon_knife_tactical
5127	0	13	weapon_knife_tactical

6.6. Required Resources

Our workflow leverages the Apache Parquet format[8] as a means to decrease system resources when storing and consuming the dataset. The columnar nature of this format allows us to:

1. Apply column-specific compression and encoding schemes, drastically reducing the required disk size to hold the dataset;

[8]`https://parquet.apache.org/documentation/latest/`

2. Read only the desired columns when consuming the dataset, again drastically reducing the amount of memory required to hold the dataset in memory.

A quick comparison between the dataset in CSV format and the currently employed optimized parquet format shows the drastic difference in disk size required to store the dataset in the hard drive (see Table 6).

Table 6. CSV vs Parquet

Unit	CSV	Apache Parquet
(GB)	83.2	9.86

7. Analyzing the Dataset

The standard way of using the dataset is to first query the index database to locate the parquet files of interest and then proceed to load the data into memory, selecting only the necessary columns to support the analysis. This workflow is very malleable, in the sense that both the SQLite database and the parquet files can be read using a wide range of libraries across several programming languages.

There are many ways one can go about using this dataset in order to gain insight into the game. To illustrate that point, Figure 3 shows a simple visualization: a contour line graph of flash-grenade explosions on every "Inferno" map – this is one of the maps in the game – played throughout the championship contained in this dataset.

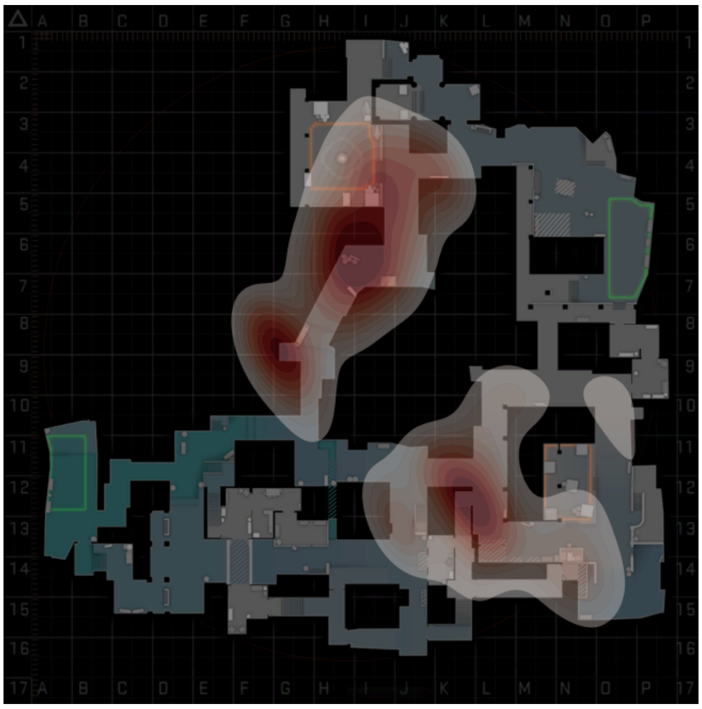

Figure 3. Contour lines of flash-grenade explosions on the Inferno map.

7.1. Further Exploration

Next, we raise some interesting topics for further analytical exploration. We refer to scenarios of interest and to research questions that can be elaborated or resolved from the availability of our dataset.

- **What is the behavior of user's mobility?**
  From the user's movement data, it would be possible to analyze how the best players in the teams move and how this affects their results. This information can be used by these same players to find their flaws and fix it or by opponents to overcome those players. *We envision that the vast literature on human mobility may be contrasted against player's mobility, and that generative mobility models can be used for practicing purposes.*
- **What are the most adopted game strategies?**
  Based on the dataset, it would be possible to categorize the different strategies used by the teams and how the teams coordinate their actions during the match. *We envision that both supervised and unsupervised machine learning tools may be used for clustering and classification purposes.*
- **What are the best strategies to win a match?**
  Based on the strategies, it would be possible to infer and compare which were used in the championship and find out which were the most victorious or most efficient against the best teams. *We envision that using reinforcement learning, one may also use our dataset for training purposes, to shed insight into novel strategies.*
- **How to identify behavior resulting from cheating?**
  Despite the great effort of championships and game developers to prevent cheating, a structured database allows anyone to carry out their search for different patterns in the behavior of a player, which could indicate some use of illegal software or abuse of some hitherto unknown bug to have advantage in the match. *We envision that statistical analysis of our dataset, together with algorithms for change point detection and outlier analysis, may shed insight into unexpected user behavior.*

8. Related work

Some of the first papers involving crawling and analysis of gaming data focused on Second Life [Varvello and Voelker 2010, Varvello et al. 2008]. Among works focusing on Counter Strike, the two more closely related to ours are [Xenopoulos et al. 2020] and [Bednárek et al. 2017]. In [Xenopoulos et al. 2020] the authors provide an open source platform to collect data from CS:GO. Their platform is complementary but different from ours. In particular, it does not bridge high-level and low-level data (see Section 3). In addition, the datasets produced by [Xenopoulos et al. 2020] are still not publicly available.

In [Bednárek et al. 2017] the authors analyze data from HLTV. Their goal is to assess and predict player performance. We envision that our work can be instrumental to reproduce and expand previous efforts such as those reported in [Bednárek et al. 2017].

9. Dataset and code availability

Our dataset is available at `https://tinyurl.com/csgodataset`
The modules comprising our crawling infrastructure are available as follows:

Crawler: `https://github.com/ErickRDev/csgo-demo-crawler`
Downloader: `https://github.com/ErickRDev/csgo-demo-downloader`
Orchestrator: `https://github.com/ErickRDev/csgo-demo-parser-orchestrator`
Parser: `https://github.com/ErickRDev/csgo-demo-parser`
Indexer: `https://github.com/ErickRDev/csgo-dataset-indexer`

10. Conclusion and future work

The increasing popularity of eSports has raised an interest in analyzing and visualizing relevant data within the domain. When it comes to CS:GO, there are websites with readily available high-level data that, when coupled with low-level data extracted from "demo files", provide the necessary input to build a robust dataset that can ultimately be leveraged to satisfy the needs of the community.

In this paper we presented such a proof-of-concept dataset along with the means by which to collect and expand on the data. Our method crawls the relevant high-level data from the HLTV website, parses the protobuf-encoded "demo files" to extract low-level data and employs a simple, yet effective strategy to bridge the gap between them.

The presented workflow represents a huge opportunity to explore CS:GO data and can be used to investigate several scenarios and aspects of the game, to create analytic reports and visualizations about the matches, players and championships.

As future work related to the generation of the dataset, we envision a number of opportunities to improve the proposed pipeline and to foster a plug-and-play analysis of the collected data. In particular, we envision additional automation of the whole pipeline, from data collection to data storage, without manual intervention.

References

Bednárek, D., Zavoral, F., and Yaghob, J. (2017). Data preprocessing of esport game records - counter-strike: Global offensive. In *In Proceedings of the 6th International Conference on Data Science, Technology and Applications - DATA, 269-276, 2017 , Madrid, Spain.*

El-Harami, J. (2015). Entertainment and recreation in the classical world—tourism products. In *Journal of Management and Sustainability; Vol. 5, No. 1; 2015.*

SuperData (2021). 2020 year in review digital games and interactive media. Technical report, SuperData, a Nielsen company.

Varvello, M., Picconi, F., Diot, C., and Biersack, E. (2008). Is there life in second life? In *Proceedings of the 2008 ACM CoNEXT Conference*, pages 1–12.

Varvello, M. and Voelker, G. M. (2010). Second life: a social network of humans and bots. In *Proceedings of the 20th international workshop on network and operating systems support for digital audio and video*, pages 9–14.

Xenopoulos, P., Doraiswamy, H., and Silva, C. (2020). Valuing player actions in counter-strike: Global offensive. In *In Proceedings of the 2020 IEEE International Conference on Big Data.*