UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ERICK ROCHA FERNANDES

EXTRACTING AND COMPOSING A DATASET OF COMPETITIVE COUNTER-STRIKE GLOBAL OFFENSIVE MATCHES

RIO DE JANEIRO
2022

ERICK ROCHA FERNANDES

EXTRACTING AND COMPOSING A DATASET OF COMPETITIVE COUNTER-STRIKE GLOBAL OFFENSIVE MATCHES

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Supervisor: Prof. Daniel Sadoc Menasché
Co-supervisor: Prof. Claudio Miceli de Farias

RIO DE JANEIRO
2022

CIP - Catalogação na Publicação

```
         Fernandes, Erick Rocha
F363e       Extracting and composing a dataset of
         competitive counter-strike global offensive matches
         / Erick Rocha Fernandes. -- Rio de Janeiro, 2022.
            34 f.

            Orientador: Daniel Sadoc Menasché.
            Coorientador: Claudio Miceli de Farias.
            Trabalho de conclusão de curso (graduação) -
         Universidade Federal do Rio de Janeiro, Instituto
         de Computação, Bacharel em Ciência da Computação,
         2022.

            1. Esporte eletrônico. 2. Conjunto de dados. 3.
         Csgo. 4. Jogos. I. Menasché, Daniel Sadoc, orient.
         II. Farias, Claudio Miceli de, coorient. III. Título.
```

Elaborado pelo Sistema de Geração Automática da UFRJ com os dados fornecidos pelo(a) autor(a), sob a responsabilidade de Miguel Romeu Amorim Neto - CRB-7/6283.

ERICK ROCHA FERNANDES

EXTRACTING AND COMPOSING A DATASET OF COMPETITIVE COUNTER-STRIKE GLOBAL OFFENSIVE MATCHES

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 16 de Março de 2022

BANCA EXAMINADORA:

[illegible]

Daniel Sadoc Menasché
Ph.D. (UMass)

[illegible]

Claudio Miceli de Farias
D.Sc. (UFRJ)

Geraldo Bonorino Xexéo
D.Sc. (UFRJ)

João Carlos Pereira da Silva
D.Sc. (UFRJ)

ACKNOWLEDGEMENTS

Meus agradecimentos vão para meus orientadores: professores Daniel Sadoc Menasché e Claudio Miceli, que no decorrer da minha graduação recorrentemente me deram oportunidades, suporte e motivação para me dedicar, aprender e é claro, por fim, concluir o curso. Adicionalmente, agradeço também aos membros da banca: professores João Carlos P. da Silva e Geraldo Xexéo cujos comentários enriquecedores influenciaram a versão final deste trabalho.

RESUMO

Há uma demanda crescente por análises mais elaboradas e significativas nos eSports: seja para o entretenimento dos espectadores enquanto assistem seus times favoritos competirem, para identificar trapaceiros automaticamente ou até mesmo para obter uma vantagem competitiva sobre um oponente, existe uma infinidade de aplicações para tais análises dentro da cena. Logicamente, segue então que existe também uma demanda por conjuntos de dados bem estruturados e organizados que possibilitem a exploração eficiente de dados e sirvam como base para tais camadas de análise e visualização. Nosso trabalho fornece os meios para a construção de tal conjunto de dados para o jogo Counter-Strike Global Offensive (CS:GO). Propomos um workflow que pode ser executado para capturar e extrair não somente metadados de torneios e jogadores, mas também os dados altamente granulares do jogo que estão disponíveis em um arquivo demofile do CS:GO. O dataset é então estruturado de forma que o metadado é exposto através de uma interface SQLite e os dados altamente granulares são armazenados em – e podem ser consumidos através de – arquivos parquet.

Palavras-chave: esporte eletrônico; conjunto de dados; csgo; jogos;

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. We propose a workflow that can be executed to fetch and extract not only metadata of tournaments and players, but also the highly-granular in-game data one can get out of CS:GO demofiles. The dataset is then structured in a way that the metadata is exposed through a SQLite interface and the highly-granular data is stored – and can be consumed from – parquet files.

Keywords: esport; dataset; csgo; games;

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

CS:GO	Counter-Strike: Global Offensive
CSV	Comma-separated Values
Ts	Terrorists
CTs	Counter Terrorists
eSports	Electronic Sports

SUMMARY

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). Thanks to the development 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 over the Internet, tournaments that until around the 2000s were mostly among amateur players started to become bigger and attracted the market's attention. Rapidly following it, a large structure was formed around professional tournaments, which became collectively known as "eSports" (a short term for electronic sports).

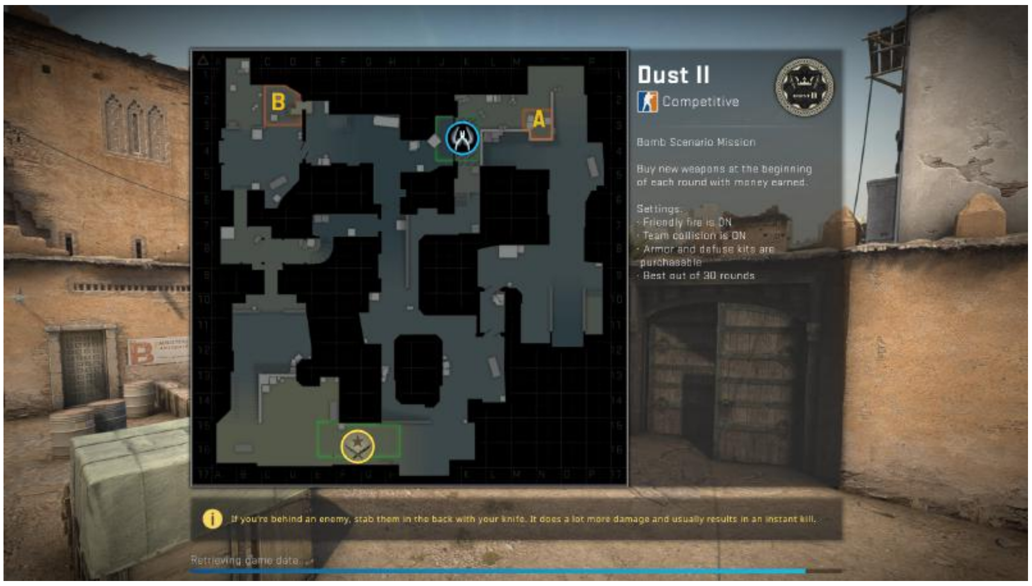
According to SuperData's annual report (SUPERDATA, 2021), digital games earned \$126.6B in 2020 and many organizations have been investing in the creation of teams and leagues to explore the competitive potential of digital games. One of the games that have for a long time been a pillar of the eSports 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.

1.1 COUNTER-STRIKE: GLOBAL OFFENSIVE

CS:GO is a first-person shooter game that revolves around a team-based action gameplay with a thematic of counter-terrorist (CTs) forces against terrorist forces (Ts). A competitive match is defined by two different teams (composed of 5 players each) playing against each other in a preselected map. A map is the virtual location where the match is played at and each map has its own particular set of characteristics and in-game topology that completely changes the match's dynamic. At the moment of writing of this article, there are 7 maps that can be selected to play in a competitive match setting: Mirage, Inferno, Nuke, Overpass, Dust II, Vertigo and Ancient. Due to the different nature of each map, the strategies employed can vary greatly and, as a result, some maps may favour a specific side (Ts or CTs). As an illustration, the Dust II map could be said to have a in-game topology that favors the terrorists forces due to how quickly this side can get to the objective sites.

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



Dust II
Competitive

Bomb Scenario Mission

Buy new weapons at the beginning of each round with money earned.

Settings:
- Friendly fire is ON
- Team collision is ON
- Armor and defuse kits are purchasable
- Best out of 30 rounds

If you're behind an enemy, stab them in the back with your knife. It does a lot more damage and usually results in an instant kill.

Retrieving game data...

Figure 1 – Dust II minimap
A loading screen showing the “Dust II” minimap.
The locations marked A and B on the map are the two bombsites.
The locations marked with a blue and yellow circles mark the CTs and Ts spawn locations, respectively.

A competitive match is divided in two halves, where a team A starts playing in the counter-terrorist side (CTs) and a team B starts playing in the terrorist side (Ts). After playing 15 rounds, the two teams switch sides (in the example above, team A goes to T and Team B goes to CT). Each round is 1 minute and 55 seconds long, with both teams having specific win conditions: the Ts 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 if not defused by the CTs. The CTs can win a round by either eliminating every member of the opposing team, by defusing the bomb after it has been planted or by staying alive in the round after the round time is over if the Ts did not plant the bomb. The first team to win 16 rounds is the match winner and the game is over.

At the beginning of each round, teams start on their own bases, which are fixed designated locations based on the selected map. Once the round timer starts to run, the goal of the CTs is to defend two key locations on the map called bombsites: the defense aims to prevent the Ts from planting the bomb on one of these locations. In contrast, the goal of the Ts is to plant the bomb on one of these bombsites and to defend it until the bomb timer runs out – and so it explodes.

An important aspect to keep in mind is the game economy: a certain amount of money is awarded to each player at the end of each round based on their win or loss conditions. The baseline gain comes from whether the team won or lost the round: winning a round grants more money to each individual player on a team then losing the round. Additional

Figure 2 – Dust II - CTs spawn location

Figure 3 – Dust II - Ts spawn location

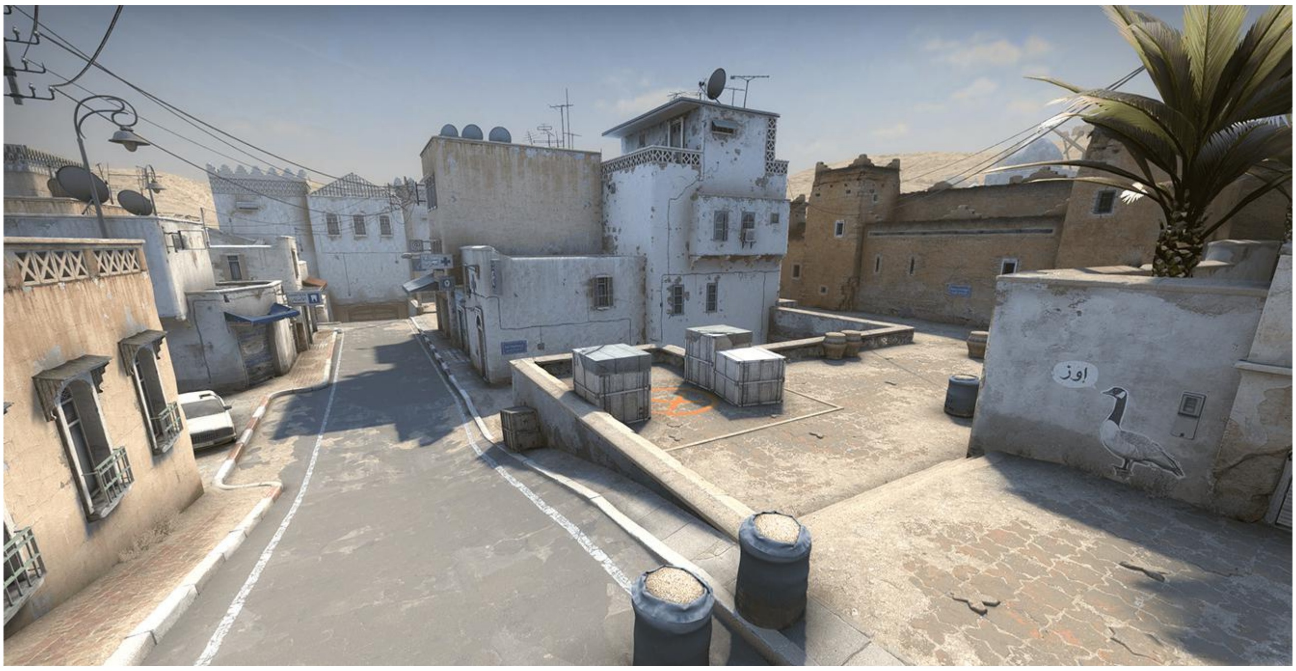


Figure 4 – Dust II - Bombsite A

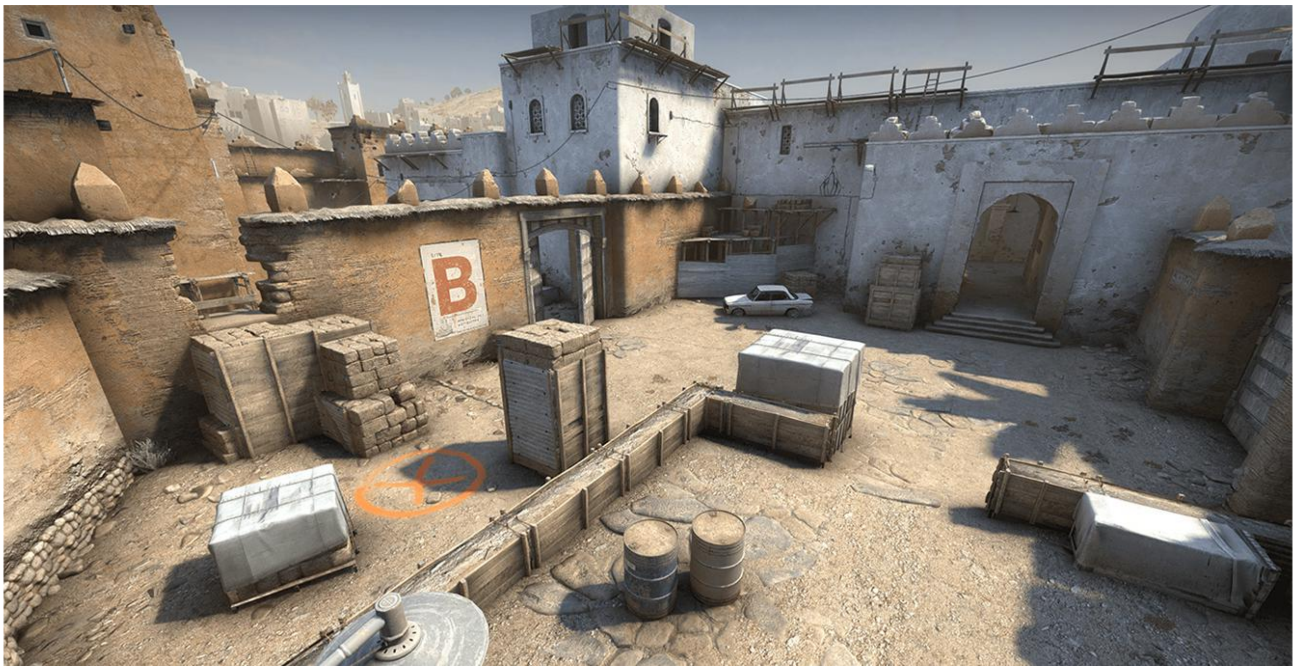


Figure 5 – Dust II - Bombsite B

bonus money is also awarded to each player based on their achievements on the round: each kill awards a specific amount of money based on the gun used, and planting or defusing the bomb also awards additional money to the player that carried out the action. This money can be used at the beginning of each round to buy guns and equipment:

- Guns can do damage and kill opponents if enough bullets land on the target. Each gun has its own set of characteristics and some are exclusive to a specific side – i.e. some guns can only be purchased by CTs while others can only be purchased by Ts. Additionally, each gun deals a specific amount of damage based on a set of parameters – e.g. the location the bullet hits the body, whether that location is protected by equipment or external environmental entities such as boxes or thin walls, etc – and better guns are the most expensive ones.
- Bullet proof vests and helmets reduce inflicted damage for each landed bullet.
- Defuse kits reduce the bomb defusal time.
- HE grenades can damage players based on the radius of its explosion.
- Flash grenades can blind enemy players for up to 4.87 seconds.
- Smoke grenades can obstruct vision of tight corridors.
- Molotov grenades can burn enemy players and cause damage.

It is clear then that players from both teams need to manage their economy in order to sustain guns and equipment that will give them a competitive edge on each round. Note that guns and equipment are carried out to the next round if the player remains alive. In contrast, if a player dies on any given round, every purchased gun and equipment is lost and needs to be re-purchased on the next round – if there is enough money available for the investment, that is.

1.2 COMPETITIVE TOURNAMENTS

There are many professional CS:GO tournaments with different formats and awards. Tournaments with higher prestige usually offers a 1 million dollar award for the tournament winner.

The most popular format is a group stage followed by playoff stage in a Swiss Format (RIVALRY, 2021). Each team plays against each other in the group stages in what is called a series. Each series is usually composed of multiple matches that are disputed in a best-of-X format (with X being the number of maps played, usually ranging from 1 to 5). Each win in a series grants a point for the winning team and once a threshold of points is achieved, the team advances to the playoff stage. The playoff stage is composed of quarter finals, semifinals and grand final. In the playoff stage, if if a team loses a series they are out of the tournament and the winning team advances to the next series.

The proof-of-concept dataset presented in this article (see Section 7) covers a tournament that follows precisely this format.

1.3 GOALS AND CONTRIBUTION

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 analytics 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. In this work, the main contributions are:

- Proposed solution to bridge the gap – between metadata and actual data – found in the literature and publicly available CS:GO datasets
- Publications (see Subsection 1.4)
- Proof-of-concept dataset (see Subsection 1.4)
- Proof-of-concept analysis (see Section 8)
- Open sourced codebase, which one can easily extend (see Section 9)

1.4 PUBLICATIONS

The work developed in this project was used to generate a proof-of-concept dataset that was published – along with a short-paper – in the DSW'21[4] symposium. A summarized description of the workflow was also published as a short-paper in the SBGames'21[5] symposium.

[2] https://sixteenzero.net/
[3] https://csgostats.gg/
[4] https://sol.sbc.org.br/index.php/dsw/article/download/17412/17248/
[5] https://www.sbgames.org/proceedings2021/WorkshopG2/219280.pdf

2 RELATED WORK

Some of the first papers involving crawling and analysis of gaming data focused on Second Life (VARVELLO; VOELKER, 2010; VARVELLO et al., 2008). Some of the challenges presented in our work, e.g., corresponding to the rate at which crawler can consume data from public sources without being black listed, are common across the works. Among the differences, we note that whereas data from CS:GO competitions is available in websites such as HLTV, Second Life by its nature counts with a distributed architecture, requiring different tools and methods for data gathering.

Interesting works can also be found in the literature on titles within the eSports scene. As a notable example, another game developed and maintained by the company responsible for CS:GO (Valve), called "Dota 2", provide equally good opportunities for data exploration and analysis, as both utilize a proprietary format to store highly granular in-game data. Works such as (YANG; QIN; LEI, 2016) and (HODGE et al., 2021) make use of this richness of data in order to tackle match result prediction, paving the way to similar works in CS:GO, such as (BEDNáREK et al., 2018) and (MAKAROV et al., 2018) – which presents game analytics for predicting winning team based on game statistics and TrueSkill.

Among works focusing solely on CS:GO, the two more closely related to ours are (XENOPOULOS; DORAISWAMY; SILVA, 2020) and (BEDNáREK; ZAVORAL; YAGHOB, 2017). In (BEDNáREK; ZAVORAL; YAGHOB, 2017) the authors report their findings on the process of parsing and analyzing CS:GO metadata extracted from demofiles – a proprietary format maintained by Valve that serializes snapshots containing highly granular data that reflects the state of the game as it takes place. Their goal is to understand the challenges faced in extracting and structuring information from this format and thus establish a strategy for acquiring quality data that can be used as a basis for assessing and predicting the performance of players and teams in CS:GO. We envision that our work can be instrumental to reproduce and expand previous efforts such as those reported in (BEDNáREK; ZAVORAL; YAGHOB, 2017).

In (XENOPOULOS; DORAISWAMY; SILVA, 2020) the authors propose a new technique for evaluating a player's impact on the results of a match. They also 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 the gap between metadata and the highly granular data one can retrieve from CS:GO demo files (see Section 4). In addition, the datasets produced by (XENOPOULOS; DORAISWAMY; SILVA, 2020) are still not publicly available.

3 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:

- Surpassing anti-crawling mechanisms while being mindful of crawling ethics, as not to disrupt the crawled source (see Subsection 3.1).
- Scaling the data fetching routine.
- Extracting information out of CS:GO "demofiles".
- Linking metadata and data (see Sections 4, 5.5 and 6).
- 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.[1]

3.1 ETHICAL ASPECTS

In order to build a proof-of-concept dataset and therefore prove the capacity of the workflow presented herein, it was necessary to acquire real data. This data was crawled from the HLTV platform (see Section 4) with due ethical concerns:

- The data acquired is public and any individual on the internet can access it without logging into the platform.

[1] Such trends also known as "meta" within the gaming community.

- The crawler – the only “closed source” code in this work – has mechanisms to make sure it acquires the data without disrupting the service, by:
  - Waiting several seconds between HTTP requests.
  - Crawling the data in a single-threaded model – i.e. no parallel requests.
- The data selected for the proof-of-concept consists of a single tournament. No more data was acquired.

3.2 UNRAVELING THE DEMOFILES

The “demofiles” (see Section 4) are protobuf-serialized[2] files containing structured information representing everything that happened throughout a match – i.e. the match can effectively be replayed by consuming the file. The problem is: the way the data is serialized is proprietary, meaning there aren’t any official sources documenting how to extract information from it. To solve this challenge without recurring to direct reverse engineering, we employed a parser[3] that abstracted away some of that work, but since the parser was still in its early days, there was still considerable need for research and a trial-and-error approach to find the desired data in the underlying protobuf schema that the parser exposed.

[2] https://developers.google.com/protocol-buffers
[3] https://github.com/saul/demofile

4 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 “metadata” – 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 simply “data” – is an actual “replay” file of a match. These “replay” files – henceforth referred to as “demofiles”, 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 the metadata and the actual data. Consider the HLTV CS:GO portal[1]: in it, one can not only browse championships and select the results of the matches played in them, but also download the demofiles of each match, which can then be consumed to extract the highly granular data representing the events played out in the match.

[1] http://hltv.org

5 WORKFLOW

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

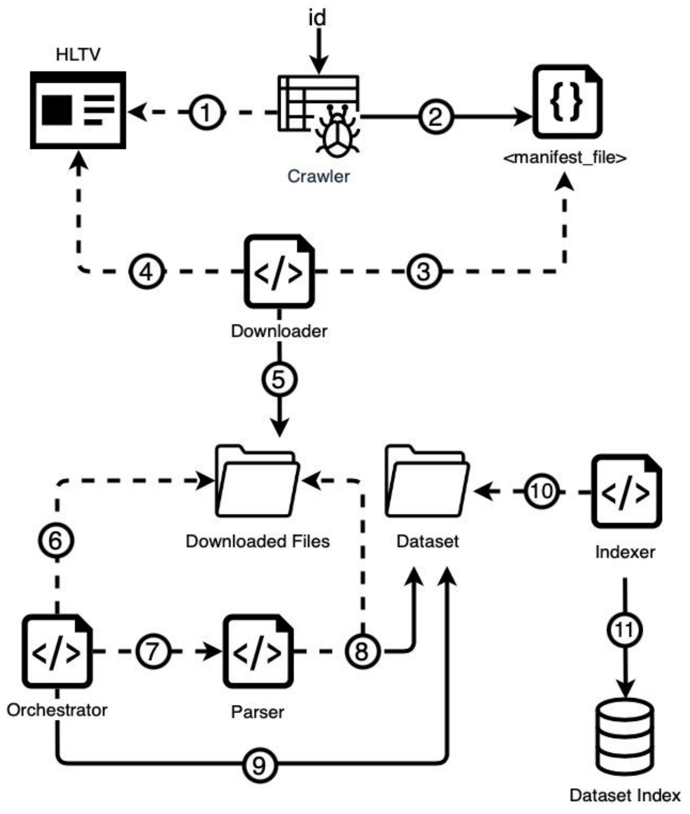


Figure 6 – Dataset generation workflow

5.1 CRAWLER

The web crawler is the entry point of the workflow: given the identification number of a tournament, 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 metadata such as the teams and players participating in the tournament, the results of each map disputed and finally the download URL for each demofile – which contains the highly granular data we want to extract.

5.2 DOWNLOADER

The downloader is the module responsible for downloading the demofiles 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.

5.3 PARSER

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

5.4 ORCHESTRATOR

The orchestrator is the module responsible for orchestrating the transformation of the demofiles into the final parquet files, which contain the highly granular data we deem important for analysis. It does this by iterating over the downloaded demofiles ⑥, invoking the parser ⑦ and finally converting resulting data from CSV to the parquet columnar format ⑨ (see Subsection 6.2).

5.5 INDEXER

The indexer is the last module of the workflow: it is responsible for bridging the gap between the metadata and actual 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 metadata data read from the manifest file and finally exposing this information through the "queryable" interface it generates: a SQLite database ⑪ (see Subsection 6.1).

6 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 that are laid out hierarchically on the local file system (see Figure 7) and is composed solely of directories, a SQLite database and a set of parquet files.

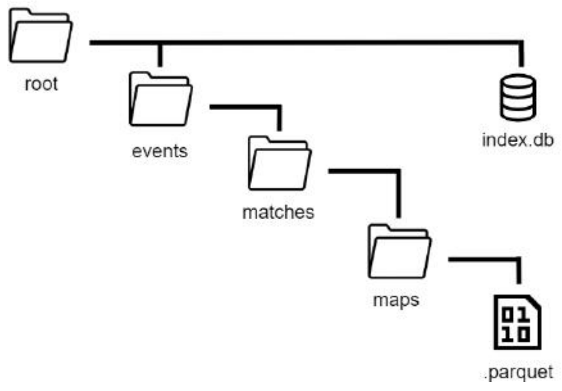


Figure 7 – Dataset Hierarchical Structure

6.1 ROLE OF SQLITE FILE

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

6.1.1 Why SQLite?

But, why SQLite? There are, of course, other options of database engines that could've been employed to achieve similar results. As a simple – and by no means extensive – list of examples, on the relational side, PostgreSQL and MySQL could've been contenders. On the non-relational side, MongoDB could've been another.

SQLite[1] is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine and the complete state of the database it generates and manages is usually contained in a single file on disk. This allows us to reap the benefits of a transactional database engine without the additional costs associated

[1] https://www.sqlite.org/about.html

with using it as a service or maintaining a remote database server. It also means we can treat the whole dataset – i.e. the entirety of the hierarchical structure presented here – as a single unit, which facilitates versioning and sharing of its contents.

6.2 ROLE OF PARQUET FILES

The parquet files contain highly granular in-game data, such as the events that occurred within the game and the positions of each player at virtually every moment of the game.

6.2.1 Why Parquet?

But, why Parquet? Again, there are other alternatives to the format that could've been employed. Obvious contenders would've been CSV or even JSON.

Parquet[2] is a storage format that allows use of compressed, efficient columnar data representation. To achieve that, it uses complex nested data structures to represent the data, leveraging the record shredding and assembly algorithm described in the Dremel paper (MELNIK et al., 2010). Effectively, it allows us to apply specific encoding schemes and compression algorithms to each column based on the data type it stores, which ultimately leads to files requiring considerably smaller disk footprints for storage and also enables programs to efficiently load the data in-memory by loading only the desired columns.

[2] https://parquet.apache.org/documentation/latest/

7 DATASET

Next, we describe the obtained dataset and dive into the contents of the highly granular in-game data extracted from demofiles. We restrict our description, focusing on a proof-of-concept of what can be achieved by leveraging the workflow described in Section 5. 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" [1] 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.

7.1 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

7.2 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.

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

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

7.3 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[2]. Table 3 shows a sample of the file contents.

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

7.4 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

7.5 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

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

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

a knife are all classified as a “weapon fired” event. Table 5 shows a sample of the file contents.

7.6 REQUIRED RESOURCES

Our workflow leverages the Apache Parquet format[3] 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;
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

[3] https://parquet.apache.org/documentation/latest/

8 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 8 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.

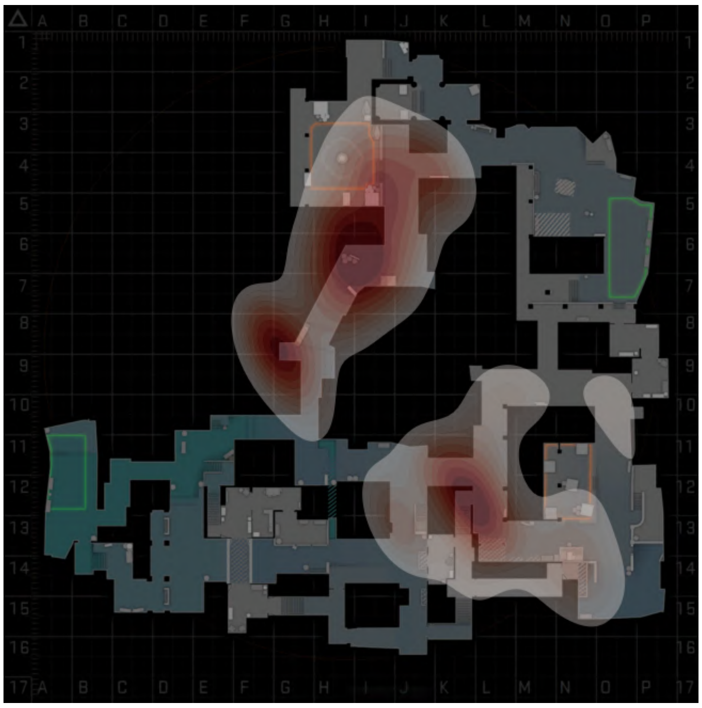


Figure 8 – Contour lines of flash-grenade explosions on the Inferno map.

8.1 EXAMPLE IMPLEMENTATION

Following is a simple implementation in Python that generates Figure 8. We start by importing the libraries we require – most of which are pretty standard when conducting Data Science in Python. Then, we proceed to populate a "Data Frame" with the data we wish to analyze. In order to do this, we leverage the local SQLite database by querying the location in the file system of the parquet files we wish to load into the "Data Frame". We can query the files of interest using metadata – for instance we can select all the games where a specific map was played. Finally, we plot the contour line graph.

Code 1 – Imports

```python
import itertools
import os
import pandas as pd
import pyarrow.parquet as pq
import sqlite3
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np
import seaborn as sns
from pathlib import Path
```

Code 2 – Setting up the Data Frame

```python
conn = sqlite3.connect("C:\\csgo-dataset\\index.db")

dfs = []
for row in conn.execute(
    f"SELECT path_on_fs FROM maps WHERE map_name = 'INFERNO'"
):
    map_dir = row[0]

    utility_file = os.path.join(map_dir, "utility_lifecycle.parquet")
    df = pq.read_pandas(
        utility_file, columns=["tick", "round", "event", "x", "y"]
    ).to_pandas()

    dfs.append(df)

df = pd.concat(dfs)
df.reset_index(inplace=True)
```

Code 3 – Plotting the data

```
img = mpimg.imread("../maps/de_inferno_radar.dds")

fig = plt.figure(frameon=False, figsize=(15, 15))
imgplot = plt.imshow(img, zorder=0)

sns.set_theme(style='dark')
sns.kdeplot(data=round_df,
            x='x',
            y='y',
            cmap='Reds',
            shade=True,
            alpha=0.33,
            cbar=False,
            antialiased=True,
            thresh=0.2,
            levels=10)

plt.show()
```

8.2 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.*

9 CODE AVAILABILITY

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 metadata that, when coupled with the highly granular in-game data extracted from demofiles, provide the necessary input to build a robust dataset that can ultimately be leveraged to satisfy the needs of the community.

In this project 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 metadata from the HLTV website, parses the protobuf-encoded demofiles to extract in-game 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. et al. Player performance evaluation in team-based first-person shooter esport. In: _____. [S.l.: s.n.], 2018. p. 154–175. ISBN 978-3-319-94808-9.

BEDNáREK, D.; ZAVORAL, F.; YAGHOB, J. 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**. [S.l.: s.n.], 2017.

EL-HARAMI, J. Entertainment and recreation in the classical world—tourism products. In: **Journal of Management and Sustainability; Vol. 5, No. 1; 2015**. [S.l.: s.n.], 2015.

HODGE, V. J. et al. Win prediction in multiplayer esports: Live professional match prediction. **IEEE Transactions on Games**, v. 13, n. 4, p. 368–379, 2021.

MAKAROV, I. et al. Predicting winning team and probabilistic ratings in "dota 2" and "counter-strike: Global offensive" video games. In: _____. [S.l.: s.n.], 2018. p. 183–196. ISBN 978-3-319-73012-7.

MELNIK, S. et al. Dremel: Interactive analysis of web-scale datasets. In: **Proc. of the 36th Int'l Conf on Very Large Data Bases**. [s.n.], 2010. p. 330–339. Disponível em: http://www.vldb2010.org/accept.htm.

RIVALRY. **SWISS SYSTEM CS:GO - HOW DOES IT WORK?** [S.l.], 2021. Disponível em: https://www.rivalry.com/pt/news/swiss-system-csgo.

SUPERDATA. **2020 YEAR IN REVIEW DIGITAL GAMES AND INTERACTIVE MEDIA**. [S.l.], 2021.

VARVELLO, M. et al. Is there life in second life? In: **Proceedings of the 2008 ACM CoNEXT Conference**. [S.l.: s.n.], 2008. p. 1–12.

VARVELLO, M.; VOELKER, G. M. 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**. [S.l.: s.n.], 2010. p. 9–14.

XENOPOULOS, P.; DORAISWAMY, H.; SILVA, C. Valuing player actions in counter-strike: Global offensive. In: **In Proceedings of the 2020 IEEE International Conference on Big Data**. [S.l.: s.n.], 2020.

YANG, Y.; QIN, T.; LEI, Y.-H. Real-time esports match result prediction. 12 2016.