Potential inaccuracies in Telraam

Péter I. Pápics, Transport & Mobility Leuven (Belgium)

August 2019

1. Introduction

Telraam has been live for a few months now, and its network of low cost traffic sensors (built on a simple Raspberry Pi computer and a low resolution camera) has collected nearly 40 million data points (as of the 20th of August) since the first test unit had sent its first data package to the server at the end of November 2018. This means 40 million objects – cars, trucks, cyclists, pedestrians – have passed in front of one of our Telraam units.

We carried out the initial calibration of the system based on data from the first three test cameras (dominated by data from the busy Diestsesteenweg in Leuven) when we had around only 1% of this data, and 1% of the number of cameras. Each camera monitors a (slightly or drastically) different street and traffic situation, therefore we expected challenges from expanding the network from a couple of cameras to a few hundred Telraam units.

In this white paper we will give a quick overview of the current measurement and classification cycle, list one-by-one the main sources of inaccuracies, and draft a solution for most of them that we will be implementing in the future over various timeframes. Understanding these challenges is a key requirement for anybody that wants to use and understand the data coming from Telraam.

2. From car to data point – measurement and classification

Each Telraam unit uses a camera to monitor the street in its field of view (FOV). The images are being processed on-the-fly, meaning that there are no actual photos or videos stored or transferred from the units. The measurement cycle has two phases, a shorter background calculation phase (when a stream of images is recorded and a median frame is calculated from them, that – in typical cases – only contains the non-variable road surface and street furniture, while all moving objects have been excluded by the median calculation), and a longer measurement phase. (We correct for this fractional downtime on the server when calculating the actual number of observations on the server.)
During each measurement phase moving objects are identified (using a simple background extraction and contour detection algorithm that separates the area – think of it as a simple shape or polygon – of an object from the background) and tracked across the FOV. For each object that meet some validation criteria (to catch outliers such as moving vegetation, slowly moving shadows, etc.) we store one line of summary information, which contains the average parameters of the given object over its visible trajectory. This summary data (mostly a set of summary data covering multiple objects observed in the measurement cycle) is transferred to the server at the end of each cycle. Since we only transfer this averaged data (and not the raw – per frame – data), the actual data packet sizes are very small.

So far we only know that we have observed something that moved along the street, but we don’t know what kind of object it was. That is where the classifier needs to step in. The classifier decides if an observed object is a car (including also vans, buses, and trucks), a cyclists, or a pedestrian.

The present classifier is a global classifier, meaning that it works the same way for each camera, no matter what the local traffic situation or the placement of the Telraam unit is. (The calculation that separates cars from larger motorized vehicles such as vans, buses, and trucks is location specific, but we will discuss this in detail later.)
The classifier was constructed from the test data of the first three Telraam units, which means that it is possibly less precise on at least some of the other cameras. Before going into detail about this, we give an overview on the ideas behind the global classifier.

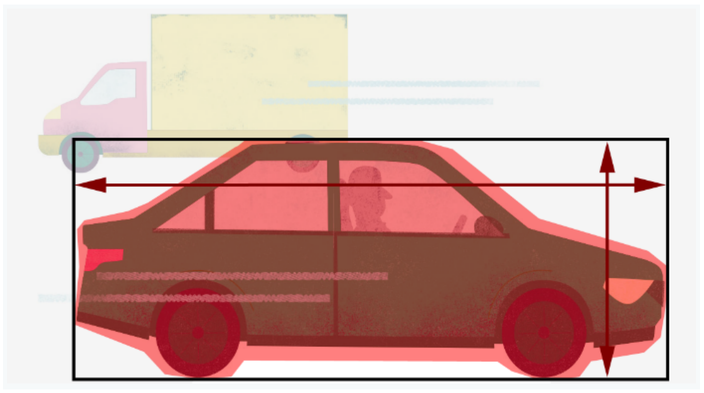
- We observe objects, that have various average properties (that are stored on the server), such as size, speed, etc. Some of these variables are distance and speed independent: such as Axisratio = height/width, and Fullness = area/(height*width), some are distance independent: such as Relativespeed = abs(speed_pixelpersec_x)/width [(pixel/second)/pixel = width/second], while all the rest are distance dependent (reminder: measured speed is distance dependent since it is in pixels per second, and objects further away travelling at the same speed will cover less pixels in the same time window): this means that the speed and size of cars/pedestrians/bikers will not be the same at different distances from the camera. We want to group (classify) the objects based on similarities in their observed properties using rules that keep being true when looking at different cameras. To make things easier, we classify traffic in the two directions (left to right and right to left) directions separately.

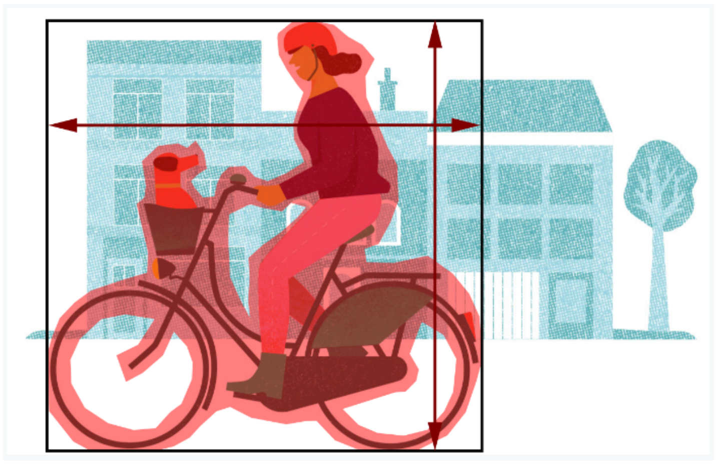
  Based on Fullness and Axisratio the 3 main classes can be relatively well separated:

  1. Cars: these are the most elongated objects, therefore have the smallest axisratio (around 0.4-0.5), also these fill up the best the rectangle fitted around them, therefore their fullness is the largest (~0.8).

  2. Bikers are more or less as wide as tall (since the length of a bike and the average height of a sitting person is more or less the same), so their axisratio is around 1, while the fullness is around 0.5-0.6, since large parts of the fitted rectangle are free space.

  3. Pedestrians are much taller than wide, so they have an axisratio typically around 1.5-2. Keep in mind that walking pedestrians actually seem wider since when the legs are open (mid step), the widest point of their body is the stride-length, which can be up to half of a normal person’s height. This mid step state also explains why pedestrians fullness is also only around 0.6, as mid step look like an upside down Y, there is a lot of space above the legs, in front and behind the torso.





To have a better grasp of what axisratio and fullness is, take a look at the figures above. The black rectangular shape is the circumscribed rectangle, while the width and the height are represented by the arrows. The area of the car or cyclists is represented here by the red shaded area. Fullness means how well this red shaded area fills up the space inside the black rectangle. Axisratio is the ratio of the two arrows. It is easy to see, that the fullness of the car is significantly larger than the fullness of the cyclist (and it becomes even more so when we are looking at the objects from the typical height of a Telraam camera), while the axisratio of the cyclists (around 1, since width and height is more or less the same) is larger than the axis ratio of the car (where width is twice the height, so axisratio = width/height = 0.5).

- The distributions of the observed objects in the various parameter spaces are most of the time not Gaussian. Also, the amount of objects that fall into the various classes/types (car/bike/pedestrian) varies on a very wide scale depending on the camera location. For some locations and directions, some object types are not even observed (one way streets for cars), some cameras see a huge number of objects, some very few.

  For these reasons, none of the standard clustering algorithms worked out well (tested KMeans, GaussianMixture, DBSCAN) for us. KMeans fails if there is not a good balance between object numbers in classes, GuassianMixtures fails as soon as there are enough objects observed that the non-Gaussian topology becomes apparent, while DBSCAN is not useful for us because it does not let you specify the number of to-be-found clusters, and its parametrization is not suitable for generalization for very different camera/traffic situations.

In practice the best properties to use during the classification is the two distance and speed independent variables: axis ratio and fullness. Not only these are great because they are speed and distance invariant, but also because the three main object types separate quite well in this 2D parameter space.

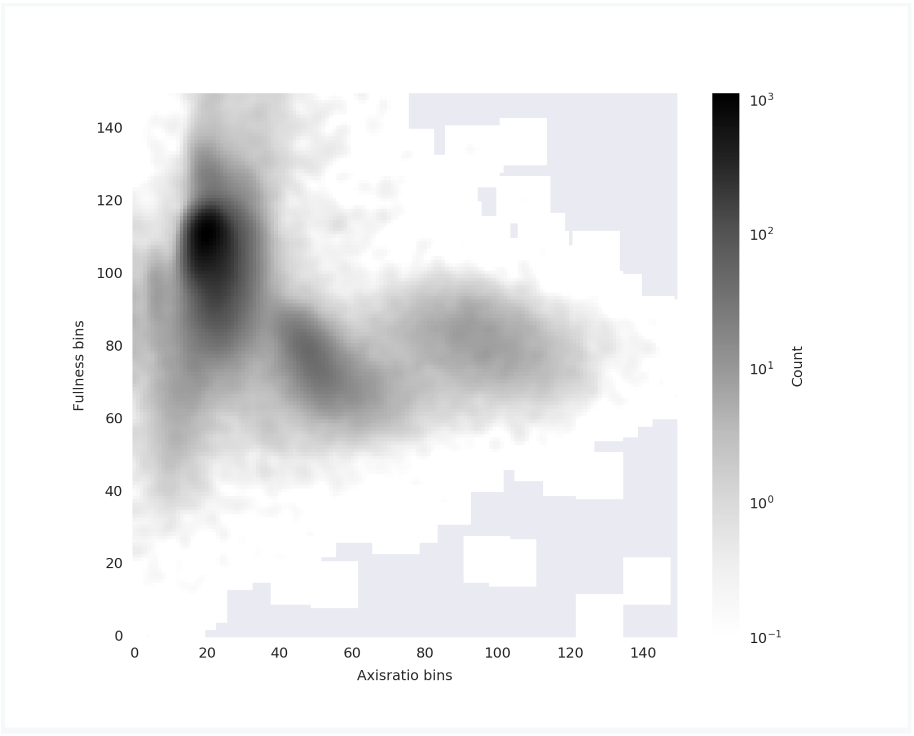
This sums up what made us consider a global classifier around the observed distributions in the axisratio – fullness parameter space.

2.2 The classifier itself

The global classifier itself is a lookup matrix that tells us for any axisratio – fullness parameter combination what kind of object we are dealing with. This makes classifying new observations extremely fast (as it is a simple lookup operation), but it also makes the classifier method rigid as new observations are not refining the classifier right now.

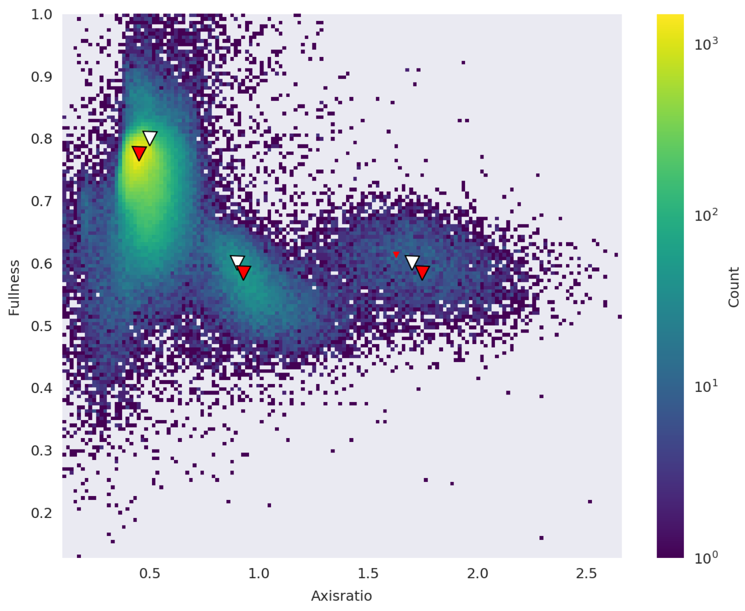
Making/calibrating the classifier therefore translates into defining the lookup matrix. We did this – again, based on the initial approximately four hundred thousand data points from the first three test cameras – following a few simple steps (for both traffic directions separately):

- We first create a 2D histogram (slightly blurred by a Gaussian filter) of the observed objects in the axisratio – fullness parameter space.



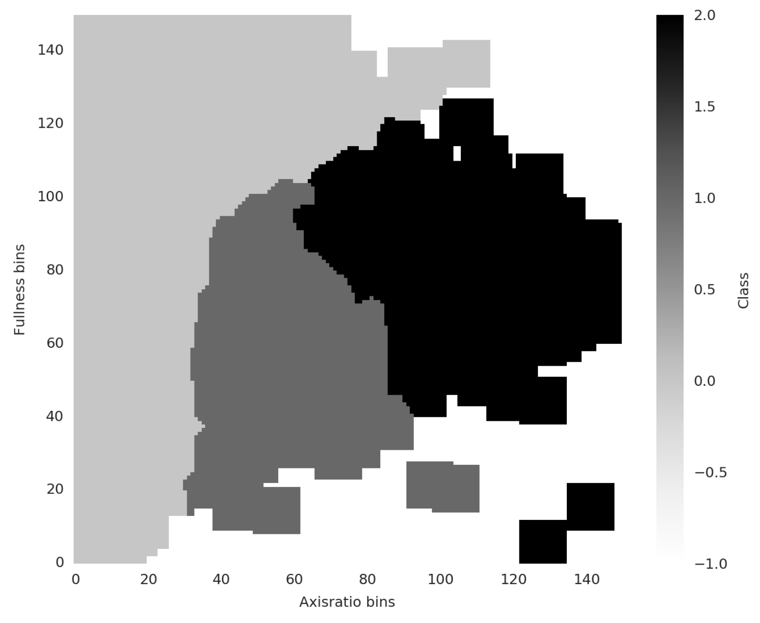
The Gausssian-smoothed 2D histogram of one direction during the making of the initial global classifier (log colouring of bin counts).

- We then use a function to find the local maxima in the 2D histogram. We select the recovered maxima the three actual ones that will match the tree classes (car/cyclist/pedestrian). The only manual supervision is that we need to know which local maximum stands for which class, but from the discussion above it must be clear by now that this is a trivial exercise.

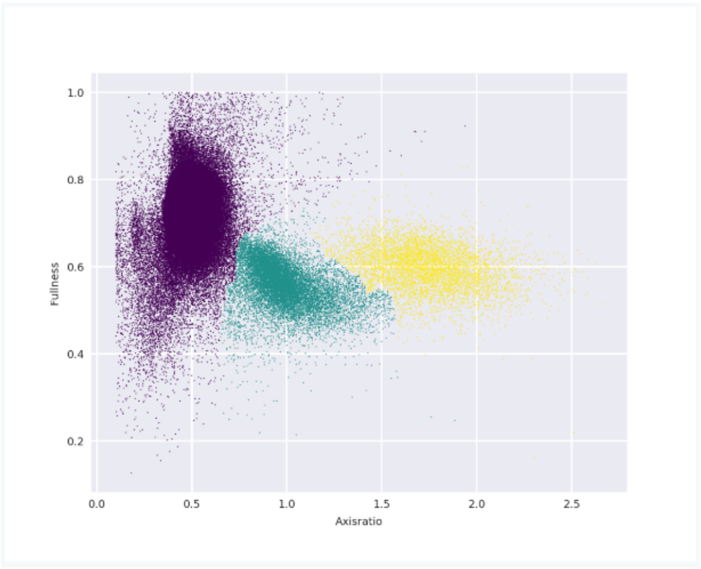


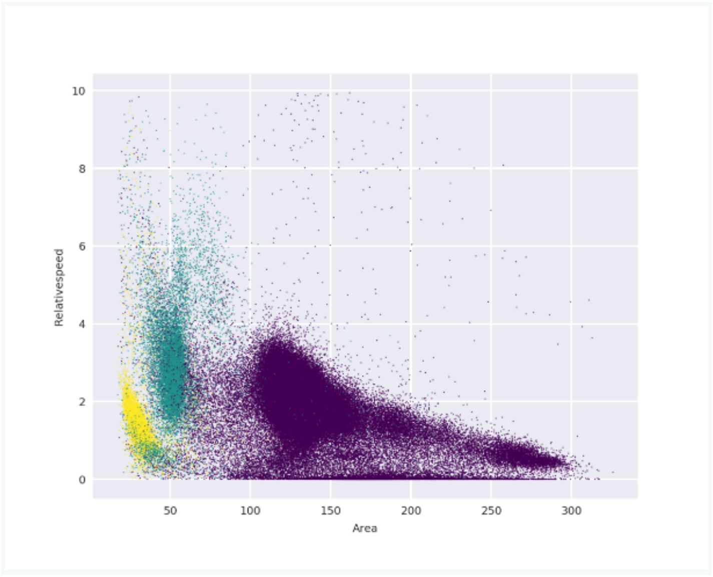
The found local maxima (large and small red triangles), the user supplied expected cluster centra (white triangles), and the final identified cluster centra (large red triangles).

- We use a hill-climbing algorithm to classify each pixel in the classifier matrix. At this point we have a smoothed 2D histogram, and for three local maxima in this histogram we have (manually) assigned labels (classes). For all non-labelled pixels in the parameter space we need to find the ‘shortest route’ to one of the three labelled local maxima. Since these maxima can be thought of as peaks in a two dimensional landscape, the task is simply to climb the closest hill following the steepest gradient until we arrive to a labelled peak. This process carried out for each individual pixel of the 2D parameter space labels each pixel according to the manually assigned class of the corresponding ‘closest’ classified ‘peak’ (cluster centrum). This completes our 2D lookup matrix that contains a class label for each combination of possible axis ratio and fullness values.



The final map – the classifier matrix – of the parameter space with cluster labels (-1 unlabelled, 0-1-2 cars-bikes-pedestrians).



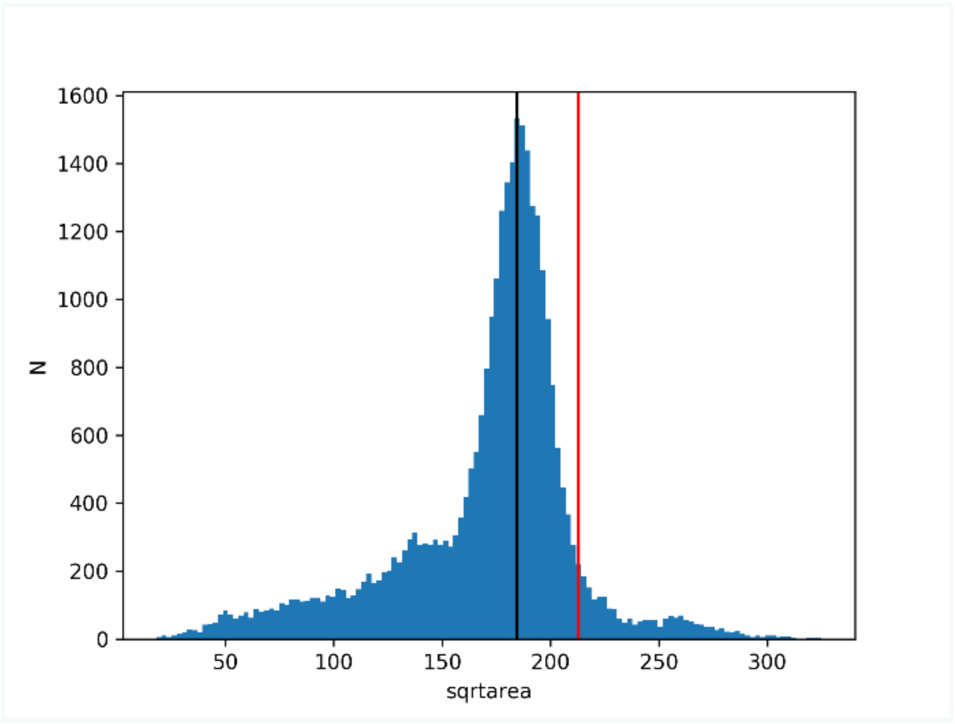


A good visual check of how well the classifier works is looking at the scatter maps of the observed objects where different colours stand for different classes. (Left panel) Not surprisingly on the axisratio fullness plane groups are separated exactly according to the classifier matrix. (Right panel) Looking at other parameters shows that distributions are much more complex (since we are not looking only at speed and distance independent variables anymore), but the coloured groups are still together, as expected from objects with similar properties. This figure shows that the observed area (here actually the square root of it is plotted) of pedestrians is the smaller, cyclists are a bit larger, while cars are much bigger with extra density groups towards the large tail of vans and trucks/buses also well visible.

2.3 Separating cars from vans, buses, and trucks

The classifier only separates cars, cyclists, and pedestrians, but we want to separate passenger cars from larger motorized vehicles. This is done on a camera by camera and direction by direction basis, using the observed parameter distributions.

The main idea is deriving a cutoff value in size that separates cars and larger motorized vehicles. Assuming that there is only one lane per traffic direction the observed area (in pixels) of cars and tracks will be distance independent (more precisely, the distance is fixed therefore the dependency does not matter anymore) for a given camera and direction, which means that there is a linear relationship between the observed and the actual size of objects. Since the class of cars is – excluding some special cases – predominantly made out of passenger cars and the number of larger vehicles is much smaller, we can assume that the main peak in the histogram of the areas amongst the objects classified as cars will correspond to the average passenger car size. Knowing the average passenger car sizes from statistics, and comparing it to the average van, and bus sizes we can derive a relative cutoff value that will stand roughly halfway between the typical passenger car and van sizes.



Example of the typical size (black vertical line), and the cutoff limit (red vertical line) derived from the histogram of objects identified as cars using the process described above for a randomly selected camera and direction. The tail of the distribution to the right of the cutoff is made up of vans, while even further to larger sizes a small peak and the distribution around contains trucks and buses. It is clear that there is a continuous transition between cars and vans, while in optimal cases a more clear separation between vans and trucks/buses.

We find that a cutoff value defined as 1.33 times the main peak in the histogram is adequate to separate these sub-classes. (The typical size is actually a projected area of the observed shape, which we calculated assuming typical camera angles using the average length, height, and width of the most sold cars, vans, and buses on the Belgian market, but we will not discuss the details further here).

3. Sources of possible inaccuracies

Now that we are familiar with the process of Telraam collecting and analysing data, we can have a look at points in the system that are susceptible to error at the current state of development.

3.1 Perceived object contours and tracking

There are situations, where the perceived contours of an object will be inaccurate.

- Shadows cast by the moving object are observed as part of the object. This is not an issue at high sun angles, or when the street is in the shade, or when the weather is overcast, or if the shadow is behind the object, only when a low sun shining parallel to the street creates long and elongated shadows over the road. This will always expand the perceived object, which will likely cause misclassification (for example cars with long shadows will be seen as vans/trucks since their perceived size will be larger than what is typical for that Telraam unit in nominal conditions).

- Objects that are overlapping from the point of the camera will be observed as one larger object, which will clearly influence the number of counted objects and their type. This way for example group of cyclists can be perceived as cars, or in the 'right' conditions with long shadows even as van/trucks.

- Specific object colours can cause sizes to be perceived smaller than they are in reality: e.g., pedestrians that wear pants which have a very similar colour to the background, will be observed less tall (therefore their axisratio will be smaller than in reality). Similarly, often car windows have the same tint (caused sometimes simply by reflection) as the road, which can result in parts of cars not being detected as part of the car. (This introduces virtual holes in the car's shape, which lowers the observed fullness value.)

None of these cases can be solved within the limitations of the current Telraam architecture. Techniques to detect shadows exist, but any processing that we do on the live stream beyond a simple background extraction and contour detection seems to put too much stress on the hardware to keep the frame rates high enough for a safe detection of faster moving objects (any typical car).

- Completely inaccurate contours whenever the traffic flow is not free (congested situations, Telraam placed very close to traffic lights): in such traffic situations the median background will not be free of objects, which will create a set of completely deformed and complex contours whenever the traffic actually moves. Since we require that a tracked object travels along a trajectory with a significant length these anomalous contours will likely not be actually counted as real objects, so we are at least not trying to classify shapes that have no resemblance to actual objects' shapes, but it means that in such situations the resulting object counts will be completely off (usually much less than what the real number should be).

- Rain falling on the window in front of the camera, or a large amount of dirt accumulating on the window (and lit by sunshine) will create either rapidly changing or blurred/low contrast images, that will not produce useful data.

We try to avoid these situations as best as we can by selecting suitable locations for the Telraam units, but we still can not exclude fully that such a traffic situation would (temporarily) occur at a few cameras. We also ask users to keep the windows clean, but stronger shower or storm might temporarily cause spurious data to be recorded anywhere.

- Tracking inaccuracies: currently there is a very simple tracking algorithm implemented, which can not handle objects seemingly merging (and then separating afterwards). This means that as long as objects do not overlap we have no issues, but if they do, counts will be incorrect. If two objects overlap at one point, then one of the objects will be lost from there on, so if they separate later on, a ‘new’ object will be formed. This would be most relevant when two cars travelling opposite directions pass in front of the camera and their contours overlap (merge) in the middle. In this case one of these cars would be counted possibly twice (if the observed trajectory of the car before and after the overlapping phase is long enough).

This can be probably solved with a more advanced Bayesian tracker, but it is not yet implemented. We estimate that this is a less significant source of error than any kind of misclassification. Also, continuous overlapping (when the objects are only observed without separation) cannot be solved by a more advanced tracker either, only by actual image recognition (by AI) which is not possible on-the-fly with our hardware.

- A large amount of contours (real or anomalous – from bad background calculation or from continuously moving vegetation, in the field of view) will significantly lengthen the time it takes for the tracking algorithm to work, which can introduce long periods of time when the camera is not observing the traffic. Normally longer downtime happens only during the background calculation and a shorter one when the tracking algorithm converts the raw frame-by-frame contour data to the simple average properties for each tracked object, but if the amount of raw contours is extremely large, then this calculation can become very long, and thus introduce and extra long downtime period. Whenever data accuracy on the Telraam website is displayed as low, it means that for one reason or another downtime for the given time interval was above the typical. The reason is in a significant majority of cases is what we just discussed.

This is a limitation of the system, a Bayesian tracking algorithm could likely handle such cases faster, but we have no tests or estimates at this point.

- Lastly, if the FOV of the camera does not cover the whole street surface, the observed numbers will be definitely too small.

We ask people to aim their cameras as good as possible (and there is an interface foreseen to help this), but sometimes – especially on streets that are very narrow – it is impossible to cover everything, for example the pavement just below the window. Our first priority is keeping the road surface in the FOV to cover the motorised traffic to the full extent, and it is possible that pedestrian and/or cyclist numbers will be underestimated in the final dataset.

3.2 Global classifier

The fact that we are using a global classifier at this point that is not camera specific has its shortcomings.

- The global classifier was constructed/calibrated from the first roughly 400000 points coming from three test cameras in Leuven. This dataset was dominated by data from one of the cameras that was situated on the busy Diestsesteenweg. While this global classifier worked very well for each of the three cameras individually (a small low traffic street with an equal amount of cars, cyclists, and pedestrians, a medium traffic street with one way car traffic and a lot of cyclists in both directions, and the Diestsesteenweg with busy mixed traffic), now we know that there are some cameras where it does not work as well. No matter how distance and speed independent the axisratio and fullness parameters are, there are situations where the typical values for the three main groups (cars/cyclists/pedestrians) shift further away from the ones defined in the global classifier. This results in misclassified objects. Possibilities are cyclist being classified as cars, or pedestrians as cyclists, and vice versa, but never pedestrians as cars or the other way around. (The only exception would be a big group of hikers or school children in a dense unresolved group, they would likely end up as a single bus/truck.)

The only possible solution here is the definition of classifiers on a camera-by-camera and direction-by-direction basis, which is currently in the experimental phase. This is a much more complex task as different cameras present us with different amount of data from different traffic situations, and often one or two of the classes are not even observed (meaning that often there are no cars in one direction, or very little pedestrians). When some classes are not present or not very well defined, then any kind of automated classifier algorithm will have difficulties finding it as a separate class. If we don’t even know in advance the number of classes that need to be found, then the task of creating a classifier matrix becomes quickly much less straightforward. Since the class of cars is in most cases very well defined, it is possible to first try to separate the class of cars, and then use another parameter, such as relativespeed, to separate cyclists from pedestrians. Even though relativespeed is not speed independent (it is distance independent), pedestrians and cyclists have typically quite constant speeds independent of the traffic situation (unlike cars that have very different speeds depending on the speed limit or the congestion level), so possibly even a global cutoff in relativespeed could work. Another option would be a more complex classifier algorithm using AI (and a larger set of parameters) but this is not yet our immediate concern.

3.3 Separating passenger cars from larger vehicles

Even assuming no misclassified objects, there are two cases where the histogram based cutoff calculation method will be less accurate than in normal cases.

- The case when there are multiple car lanes in one direction. Multiple lanes means there is not one typical distance between the moving cars (including vans, trucks, buses) and the Telraam camera, meaning that the size histogram will actually have multiple peaks. (To understand this easier, imagine there are two traffic lanes, and uniform sized cars travel along the centreline of these lanes. The camera will see two object sizes, each specific for one direction. This results in two peaks in a histogram, broadened by the actual size distribution and the fact that cars do not exactly drive over the centre of the lane, etc.)

This is a situation that is quite hard to handle. We could try to separate the observed objects based on their average observed mean position (‘centre of gravity’) in the vertical direction (as lanes are separated this way), and then do a cutoff calculation lane-by-lane, but the scatter is likely to be large enough that this method would not be foul-proof either. (Since trucks have their centre of gravity higher than cars – as they are taller –, from the point of a Telraam camera a truck in the near lane might have a centre of gravity exactly at the vertical position where the centre of gravity of a car in the far lane would be… So we would introduce another level of complexity with a new level of potential ambiguity.)

At this moment the cutoff is calculated from the peak position in the histogram, but we are extending the calculation to actually use the average of the left and right half maxima around the peak (the centre point of the full width at half maximum). This slightly improves the situation in many cases, but when there are two lanes in one direction, this still causes a too small cutoff for the lane closer to the camera, and a too large for the one further away.

- Streets that are one way for motorised traffic, and that one way is only allowed for buses are completely incompatible with our cutoff calculation method, since it assumes that the dominant object size in a direction is the size of a passenger car.

There are no streets like this with a Telraam camera right now, but this is definitely something to keep in mind, and something that cannot be solved automatically.

- Already mentioned above, that shadows can cause object misclassification, this way for example a group (or even two cyclists in the ‘right’ conditions of shadows and overlap) of cyclists can be classified as a car. If this happens in a street that is one way or not even allowed for cars, then the cutoff calculation will happen on a size distribution that does not even contain real cars, only misclassified ones, meaning that the cutoff size will be in the size range of cyclists (since these are the misclassified objects), so a large amount of these objects will end up with a van/truck label, while actually they were cyclists.

It is not yet implemented, but we will tackle this by looking at the shape of the histogram. We have seen that where this happens the histogram is very skewed, while in normal cases it is much more symmetric. So based on the shape of the histogram we can detect the anomalous cases, and for these cases we will simply say that no objects can be trucks. This way groups of cyclists might still end up as cars, but cars that drive in an illegal direction will still be classified as cars. An other option would be to catch these anomalous one way streets based on the histogram shape and then downgrade all cars to cyclists assuming that all object that show up as cars are just misclassified cyclists. Unluckily this way we would also not be able to catch people actually driving in the illegal direction with a car, so there is no perfect solution. Right now we are aiming for the first option.

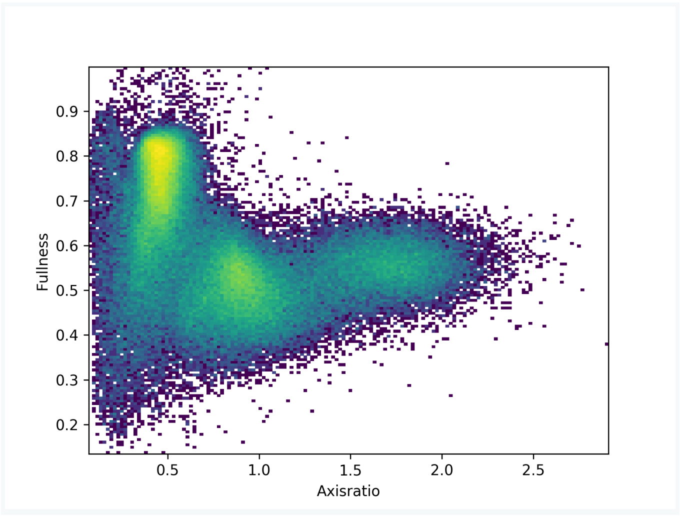
- It needs to be noted, that the border between cars (up to and including SUVs), and vans in size is not very well defined, a larger SUV and a smaller van are quite close to each other in their dimensions. The distinction between vans and buses/trucks is much more clear, as the gap between them is much larger. So while there is a more or less continuous transition from SUVs to vans, there is an empty space on the size scale between vans and buses. Still, separating personal vehicles from delivery vans, trucks, and public transport vehicles is more interesting, therefore the chosen limit between cars and vans. Furthermore for streets where the Telraam unit is close to the road surface (most urban streets are relatively small and narrow in Belgium), buses and trucks are too long to fit fully into the FOV, which means that their perceived length (and area) is often smaller, so a cutoff value derived from manufacturer data would not always work should we want to make the distinction between vans and trucks/buses instead.

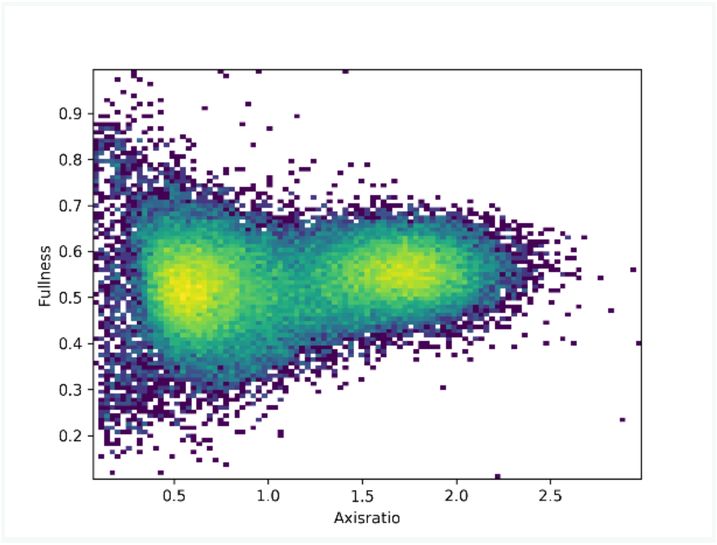
For these reasons the separation of motorized vehicles into passenger cars and larger objects should be taken more as an estimate (which can still be very good at times) than as a direct measurement.

3.4 Example of the global classifier failing and incorrect car/truck separation

In the example below we show how all the aforementioned potential pitfalls materialise themselves in the case of a suitably selected actual Telraam camera. This Telraam unit is chosen to represent the smaller portion of units where things actually go wrong, and thus it is not representative of the typical, average situation.

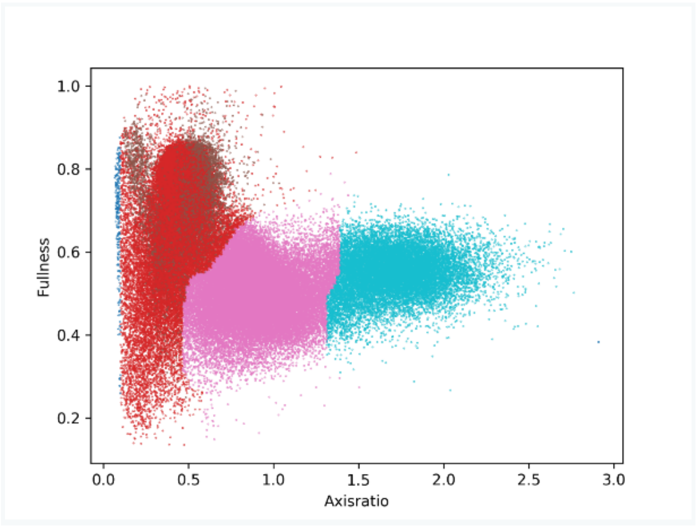
First, let's have a look at the distribution of objects observed in both directions of this selected road segment.

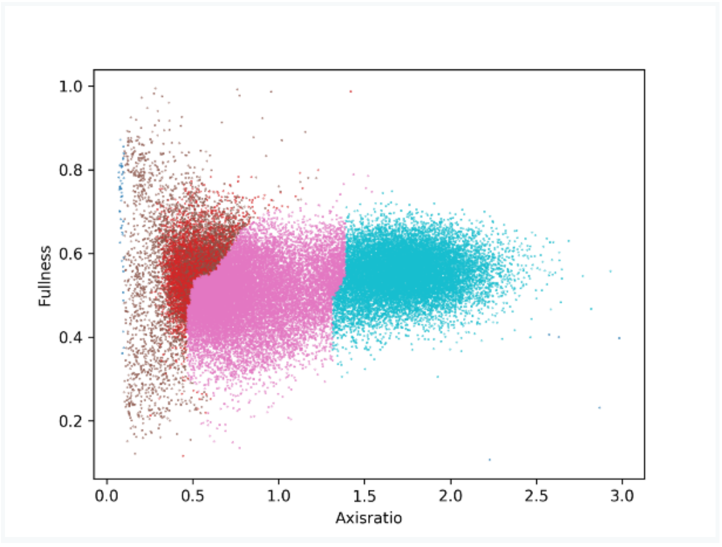




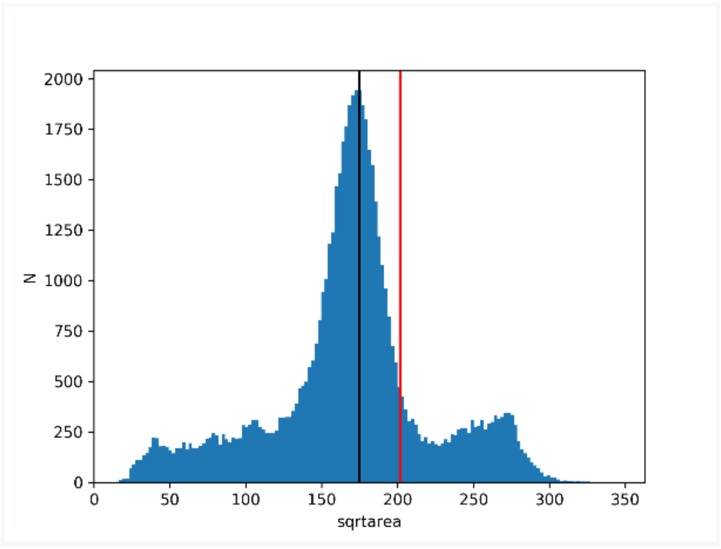
The two figures above show the 2D density histogram of observed objects by the same Telraam camera, but the left and right hand side panels show objects travelling in different directions. Colours are logarithmic, so the actual condensation of different groups is much stronger than it is suggested by these figures (dark blue squares correspond to orders of magnitude less object than the yellow squares.). On the left all road users are observed (we refer to the similar figure in the global classifier section), with the most prominent being the group of cars near the top left corner of the figure. This group of cars is clearly missing on the right hand side figure, which tells us that this street is one way in terms of car traffic.

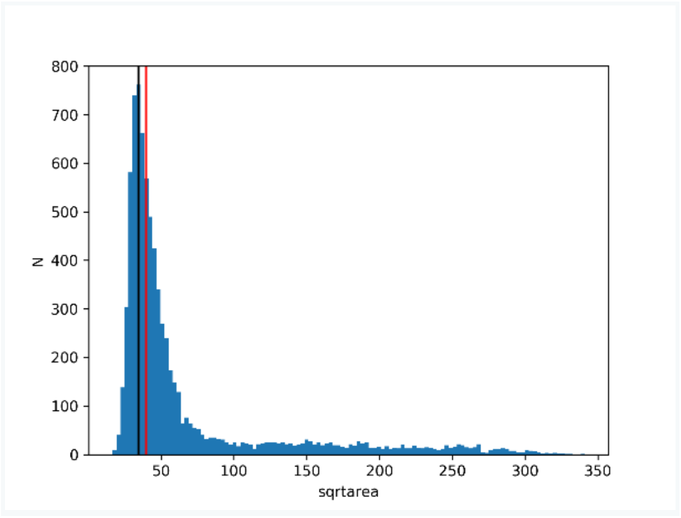
The problem is that for this specific Telraam unit (for one reason or another) the global classifier does not work very well, and especially for the direction where the traffic is not allowed for cars (right hand side figures) it misclassifies a lot of cyclists as cars (some misclassification can also be seen on the left hand side, but to a much less significant extent). On the figures below different colours stand for the different classes (red and brown are cars and trucks, pink are cyclists, and blue are pedestrians – according to the global classifier).





The figures below show the cutoff calculations for the class of cars. On the left hand side everything looks correct, there is a distinct distribution for cars and even a nicely visible second peak at larger sizes corresponding to trucks and buses. On the right hand side however things are not correct. Since a lot of cyclists have been mis-classified as cars, the huge peak in the distribution is actually the typical size of these mis-classified cyclists, therefore the derived cutoff is completely incorrect and unsuited to make a distinction between cars and trucks. It is likely – as explained earlier – that none of these objects is an actual car or truck.





These kind of issues can only be solved by creating classifier matrices for each camera and direction, and also by detecting non-represented road user classes in data sets (so here automatically detecting that the group of cars is not present on the right hand side – either already during the classifier definition, or, preferably, during the cutoff calculation based on the very typical skewed histogram).

4. Conclusions

We have given an overview of the current state of the data analysis and classification part of the Telraam system as it stands at the end of August, 2019.

The system uses techniques that enable real-time data processing without the need of storing actual images, and which minimise the size of data packets that need to be transferred to a server. Inherent to this design (and the current state of code development) there are some challenges and limitations that need to be considered when looking at data of individual Telraam units.

While in average weather and traffic conditions the system works with good precision, we have listed various situations where the quality of one or more sub-process can (temporarily or consistently) deteriorate. Some of these shortcomings can be solved by further developing, e.g., the classifier code, while some are inherent to the design (often forced on us because of hardware limitations or privacy considerations).

Finally, we need to note that a full validation of the results has still to be carried out, but in the meantime simultaneous traffic counts by the police and some citizens have already provided ample evidence that we are achieving very good precision (most of the time), especially in terms of car (including vans, trucks, and buses) numbers – see the appendix for a few examples. Unluckily the limitations discussed above mean that at this there is a chance that we are underestimating the share of sustainable traffic. (The most important reasons: pedestrians usually not being seen just under the camera on the near-side of the road, and – multiple – cyclists being misclassified as cars.)

For such a simple and low cost system these results are more than promising, especially when properties like cost/data point, temporal resolution, uptime, maintenance, etc. are taken into account. Deploying a network of 100 cameras in a small city could provide continuous (in daylight hours) traffic monitoring for the fraction of the costs of any other method.