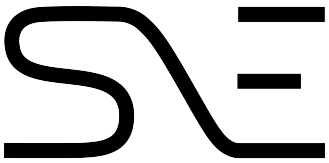

System and Network Engineering

Plug-and-play Raspberry Pi-based video filtration system:

A novel approach to real-time on the wire reversible PII anonymization in video streams using commodity hardware

Chris Kuipers* Swann Scholtes†
*chris.kuipers@os3.nl, †swann.scholtes@os3.nl

Institute of Informatics
University of Amsterdam
February 11, 2018

***Abstract*—With Video Surveillance Systems becoming a commodity to our modern society, the call for preserving our privacy within these systems is evenly getting stronger. The systems that are available today lack certain aspects that render them useless to ordinairy businesses and organisations. In our novel approach filter Personally Identifiable Information (PII) from real-time camerafeeds, we present a plug-and-play solution, based on commodity hardware, that can be used existing Video Surveillance setups. Our proposal makes use of a Deep Neural Network (DNN) to identify regions of interests (ROIs), and can blur them using three anonymization filters. The original video stream is AES encrypted before storage to allow access under certain circumstances later on. We tested various configurations on four different hardware setups to derive the feasibility of our proof of concept on commodity hardware. We proved that not all hardware platforms are equally capable of outputting a workable video feed. Our results show that the Raspberry Pi 3 is outputting a stuttering video, that is comparable to traditional video surveillance systems.**

I. Introduction

Nowadays, Video Surveillance Systems are employed in a number of wide-ranging industries, from healthcare and assisted living communities to retail and hospitality. Each industry is using these systems for a variety of applications.

Back in the days, Video Surveillance systems were operated by humans who monitored the camera feeds[1]. An increasing amount of technology is used to automate the process of classifying events and automatically recognizing oddities and accordingly[2]. This is done by identifying anomalies using complex probabilistic calculations and visual analysis[1]. The methods of using smart, automated surveillance systems are to be considered heavily privacy invasive. The type of systems can track all persons and objects in its view, threatening fundamental human rights along its way[3].

In health care, Video Surveillance solutions aim to aid the recovery process of patients or improve the quality of life of the elderly[4]. In such a solution, cameras monitor the patients. While this instrument could have a significant positive impact, the cameras are located in the homes and living areas of those patients. Therefore, the privacy impact of the Video Surveillance is even more substantial than the impact of Video Surveillance in the public domain. While the privacy of those people are affected the most, the privacy issues of Video Surveillance is not limited to health care alone. Also in other situations, Video Surveillance systems were used for monitoring certain trends and anomalies[5, 6].

Most stages in the solution we propose are not new. A lot of work has already been done in classifying PII, identifying ROIs via the use of Deep Neural Networks (DNNs) and (ir)reversibly filtering the data. The solutions that are on the market today

are neither based on commodity hardware, resource efficient, effective or open source.

In our approach to protect the privacy of people, we are aiming for a solution that filters personally identifiable information (PII) from live camera feeds, by identifying and obfuscating regions of interests (ROIs). The solution we propose is deployable in a plug-and-play manner in existing Video Surveillance setups through the use of commodity hardware. We present the solution in the form of a proof of concept, where the code is open source and publicly available under the GPLv3 license. We test our setup on four different hardware platforms to measure the impact of various configurations. Based on these results, we can derive the feasibility of our setup on commodity hardware.

II. Research Questions

To achieve our goal, we deduced the following research question:

- How can commodity hardware be used to filter Personally Identifiable Information from real-time video streams?

Based on that question, the following sub-questions were deduced:

1) What types of Personally Identifiable Information are of interest that need to be filtered from the camera feeds?
2) What feasible anonymization techniques are available to reversibly anonymize regions of interests?
3) How can the proof of concept be tailored to commodity hardware?

III. Related Work

The implications of Video Surveillance systems to privacy has already had quite some attention, shown by numerous papers who try to solve this exact issue, all using their own approach.

The first part we try to achieve is to identify regions of interest (ROIs) that contain personally identifiable information (PII) in a video stream. Research has already been conducted regarding classifying PII.

The second part we try to achieve is filtering PII by removing the information from ROIs. Other researchers have already looked into the challenges regarding anonymization techniques and the aspects regarding their reversibility and security.

A. Personally Identifiable Information

Before we can identify ROIs, we first need to define what types of information need to be considered as *Personally Identifiable*. The define Personally Identifiable Information as *any information related to a natural person or 'Data Subject', that can be used to directly or indirectly identify the person*[7]. Although this definition is quite clear, it is also quite broad. To put this into context regarding the filtration from video streams, the following concrete examples can be derived[8]: *Personal attributes*; such as face and other distinguishing characteristics and specific behaviour. *Information on paper*; such as identification documents, banking account numbers and addresses. *Information on other media* such as on smartphones and on computer screens.

B. Approaches to reversibility

For numerous purposes, like court-orders, there needs to be a way to access the original video stream. Just anonymizing the video feed, without any means to recover the original information it contained, is not enough. In practice this can be solved in multiple ways;

- One-way method: Irreversibly altering the stream and saving an original copy;
- Two-way method: Altering the video stream in a reversible manner (two-way function);
- Splitting up the stream in multiple chunks.

1) One-way method: Wickramasuriya et al. describe a method where they use a one-way anonymization function and store both the anonymized and the original video stream[9]. The original stream being encrypted before storage. In normal operation, only the edited, anonymized, version is used. In special cases, when the unfiltered version of the video feed is needed, can be accessed by whomever holds the encryption key.

2) Two-way method: Another solution is the use of a two-way anonymization method. The original information is scrambled in such a way that the process can be reversed, for example by encrypting the ROI with a shared secret. Pantoja et al. reviewed a bunch of methods, both permanent and reversible, and compared them to eachother.

3) Splitting up: In their paper, Upmanyu et al. describe an alternative approach to privacy preserving surveillance systems. Instead of altering the original video stream by filtering out or encrypting ROIs, they

proposed a solution where they split the stream into smaller parts or chunks to be processed separately.

C. Level of anonymity

The level of anonymity the filtration process offers, all comes down to the effectiveness of the anonymization algorithm being used. The three approaches to reversibility have their own set of characteristics.

1) One-way method: Ruchaud and Dugelay demonstrated a way to detect whether images have been altered using a privacy filter. In case the images have been altered, they were able to reliably tell which filter had been used. They proved that it is possible to reverse the filter process to such extent that automated facial recognition is possible again. Worth noting is that they conclude that although numerous privacy filters appear to successfully hide Personally Identifiable Information, those filters may not be as reliable and secure after all[12]. Ruchaud and Dugelay particularly call out the techniques *blackener*, *normalized box pixelization* and *(Gaussian) blurring*. However, it is worth mentioning that they altered an image test set using the Gaussian blur standard deviation parameter ranging from 2 to 8 and the pixelization parameter size, ranging from a 3x3 to 10x10 pixel grid.

2) Two-way method: The method proposed by Rahman et al., based on Chaos Cryptography is an example of such a solution that has been reviewed[13]. Their proposal supports multiple levels of reversibility; authorization levels can be defined, so that only certain types of information will be reversed, while other types of information will still be kept hidden.

Two other reversible methods reviewed by Pantoja et al. are warping and pixel relocation. Korshunov and Ebrahimi describe a warping method that obfuscates faces in videos and images[3]. Using their method, faces can be warped according to various key points. The security of the method relies upon the level of warping that is applied. They claim that the warping process is secure, because their algorithm makes use of a shared key and a transform matrix. They do not use proven and well known symmetric encryption algorithms so the level of security offered by the warping method is questionable. This same issue also arises in the method [14] published about[14]. They proposed a method that relocates the pixels in a ROI according to a fixed pattern. This same, fixed, pattern is used in all operations. Anyone that knows the pattern can undo the filter and obtain the original version. Therefore, this method is far from secure, as proved by Pantoja et al.[10].

3) Splitting up: The solution Upmanyu et al. propose, relies upon the assumption that *"each image by itself does not convey any meaningful information about the original frame, while collectively, they retain all the information"*[11]. Therefore, they claim that their solution is both privacy preserving and efficient, because there is no need to edit the stream.

IV. Methods

The test environment consists of an Ubiquiti UVC-G3-Dome IP camera, router and an interception device. The interception device is a piece of hardware that contains the OpenCV3.3.0 software library to be able to process the video stream generated by the IP camera. This setup is shown in Figure 1. The IP-camera streams the video as an H264 encoded RTSP-stream, in Full-HD (1920x1080) at 30 fps.

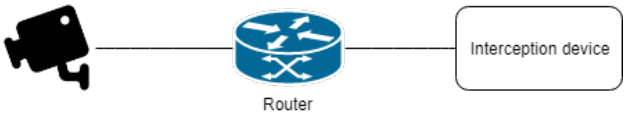

Figure 1: Setup overview

To determine the influence of the hardware on the setup, different hardware platforms will be used. The four different setups are based on a Dell desktop, Intel Atom server, Intel Atom laptop and a Raspberry Pi 3. The exact hardware specifications can be found in the hardware Appendix IX. Each batch of tests will be run on the different hardware platforms.

The project's source code is fully open source and publicly available under the GPLv3 license on our GitLab page[15].

OpenCV includes a module to detect objects using pre-trained deep neural networks. The deep neural network needs model data to detect objects. The training of a DNN model is outside the scope of this project. Because we are interested in detecting multiple types of PII, pre-trained models that only detect humans are not sufficient.

For our proof of concept we set out to find a model that not only detects people, but at least one other type of PII we defined, based on the findings in the related work section. We ended up with using the publicly available Caffenet Caffemodel described by

Donahue[16]. This model is trained to detect 20 types of everyday objects, including person and monitor screen object detection. To proof that our concept of filtering multiple types of PII works, this model sufficed. The accuracy of this model is included in Appendix X.

A. Unit of measurement

Each experiment will run for 15 seconds and is repeated 10 times giving a minimum of 140 results per hardware platform. We measure the frame throughput in frames per seconds (fps) on the different hardware platforms to determine the performance hit of each test. A median will be calculated from each individual test giving the average of each test conducted.

B. Measurement technique

In a normal setup, the interception device will process the images according to a set rate, this is either the same fps as the video source, or a statically configured value.

In our setup we intend to measure the maximum performance of each hardware platform under given situation. Therefore, we did not specify any fps rate beforehand. As a result, each device will try to process frames as fast as possible from the IP camera video feed. If the interception device has enough resources to handle more frames than the IP camera is able to put out, it will take the same frame multiple times and treat each frame as a new frame. In the same way, when the interception device lacks the resources to process the video stream at the frame rate the IP camera is putting out, the frames will be dropped. As soon as it can process the next frame, it will take the most recent one in the video stream.

Therefore, this setup ensures us that we measure the maximum performance of the interception device under the different circumstances.

C. Experiment layout

We use a static setup pointing the IP camera at 2 monitors, to ensure that between the tests, the same number of objects are being detected. This eliminates the influence the amount of regions being detected have on our results. This setup is shown in Appendix XI

The proof of concept includes 14 different tests that measure the performance of the interception devices under various loads.

The first test will serve as a **baseline**. The baseline will be used to see what impact the other tests have to the performance of the interception device. The baseline test picks up the video feed and discards it afterwards. It does not perform any manipulation on the feed.

The **detection** test will use the OpenCV Deep Neural Network in combination with the Caffenet Caffemodel to detect monitors and persons, though in our static setup, we will only detect monitors.

The **boxes** test will draw rectangle boxes around the detected objects.

The **label** test will put descriptive labels on the detected objects, stating what kind of object has been detected alongside the confidence level expressed in percentages the Deep Neural Network has that it is actually that particular object.

The **save edited stream** test archives the modified stream to storage in H264 format so that it can be re-watched later in time.

The **save original stream** test archives the original stream in H264 format to storage so that the original unmodified stream can be re-watched later in case of a court order.

Based on the findings in the related work, we decided to cover the reversibility aspect of our setup on the container-level. Therefore, we encrypt the original stream. We test the perfomance of this step using the **encrypt original stream** test. This test encrypts the original video stream before it is stored or transmitted over the network by using AES-128-CBC.

The **re-stream** test broadcasts the anonymized video stream just like the original IP camera does. To accomplish this, the test uses *ffmpeg* and *ffserver*.

The **blur** test applies the pixelation blur anonymization technique to the detected objects. The blur level used in this test is the maximum value accepted by OpenCV, which is 50. This translates to a square of pixels of 50x50 pixels.

The **blur and padding** test applies 25 extra pixels around the detected objects making the region of interest bigger. The extra padding is added to determine how the size of the ROI affects the performance of each blurring technique. The padding is useful if objects move faster than the DNN can process. In that case, objects could move outside of the blurred region and unveil its contents.

The **Gaussian blur** test applies the Gaussian blur technique to the detected objects. The standard devi-

ation used in this test is 25, which is the maximum value accepted by OpenCV.

The **Gaussian blur and padding** test applies 25 extra pixels around the detected objects making the region of interest bigger. This is done to see how the region of interest size effects the Gaussian blurring technique.

The **masking** test applies the masking anonymization technique to the detected objects in the video stream. Masking a ROI means that the area will be filled with a solid color, usually black.

The **masking and padding** test applies 25 extra pixels around the detected objects making the region of interest larger. As a result we can measure the impact the size of the region of interest has on the masking technique.

D. Cumulative testing

The next step is to bring in logic to the tests being conducted. It would not make sense to draw boxes or put labels on the detections if the detection test itself is not being conducted. Table I shows an overview of how the different tests are constructed. Up until test 8 (re-streaming), all tests are cumulative. Starting at test 9 the actual anonymization techniques will be applied.

	Baseline	Detection	Boxes	Labels	Save anonymized stream	Save original stream	Encrypt original stream	Re-stream	Blur	Blur + padding	Gaussian blur	Gaussian blur + padding	Masking	Masking + padding
Test 1	✓													
Test 2	✓	✓												
Test 3	✓	✓	✓											
Test 4	✓	✓	✓	✓										
Test 5	✓	✓	✓	✓	✓									
Test 6	✓	✓	✓	✓	✓	✓								
Test 7	✓	✓	✓	✓	✓	✓	✓							
Test 8	✓	✓	✓	✓	✓	✓	✓	✓						
Test 9	✓	✓	✓	✓	✓	✓	✓	✓	✓					
Test 10	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓				
Test 11	✓	✓	✓	✓	✓	✓	✓	✓			✓			
Test 12	✓	✓	✓	✓	✓	✓	✓	✓			✓	✓		
Test 13	✓	✓	✓	✓	✓	✓	✓	✓					✓	
Test 14	✓	✓	✓	✓	✓	✓	✓	✓					✓	✓

Table I: *Shows how the different tests are constructed*

V. Results

The results of all 14 experiments on the four hardware platforms are shown in Table II. The unit of measurement is frames per second (fps).

	Dell Desktop (fps)	*Raspberry Pi 3 (fps)*	*Intel Atom Server (fps)*	*Intel Atom Laptop (fps)*
Test 1	120.1	9.2	7.9	3.46
Test 2	118.4	9.1	7.8	3.42
Test 3	117	9.1	7.8	3.45
Test 4	115.1	8.8	7.6	3.42
Test 5	101.1	7.3	6.2	3.06
Test 6	82.5	6.4	5.3	2.91
Test 7	85.8	6.4	5.3	2.92
Test 8	85.3	6.4	5.3	2.96
Test 9	76.6	6.0	5.4	2.87
Test 10	78.6	6.2	5.3	2.82
Test 11	57.9	4.1	4.78	2.5
Test 12	53.6	3.8	4.79	2.58
Test 13	82.0	6.3	5.4	2.88
Test 14	82.0	6.3	5.3	2.91

Table II: *Shows the relation between the baseline measurement and the performance hit in frames per seconds*

The results from the 14 Dell desktop experiments are plotted in Figure 2. Test 1 shows that the baseline measurement for the Dell desktop is 120.1 fps. Test 2 shows that detecting objects with the deep neural network model results in a 118.4 fps. This is 98.58% compared to the baseline. After adding rectangle boxes on the detections, test 3 shows that the frames per seconds achieved is 117. This is 97.42% of the baseline measurement.

Test 4 also puts labels on the detection, which gives a performance of 115.1 fps. Compared to the baseline measurement this is 95.84%. Test 5 also saves the anonymized video stream to disk, which performs with 101.1 fps. This is 84.18% of the baseline measurement. test 6 shows that 82.5 fps is achieved while also saving the original video stream to disk. 82.5 frames per second is 68.70% compared to the baseline. Also encrypting the original video stream yields 85.8 fps as shown in test 7. This is 71.44% compared to the baseline measurement. Test 8 shows that at the same time re-streaming the anonymized video stream performs with 85.3 frames per second. 85.3 frames per second is 71.02% compared to the baseline. Anonymizing the detections in

the video stream with a blur yields 76.6 fps as shown in test 9. That is 63.78% compared to the baseline. Test 10 shows that 78.6 fps can be achieved while also adding 25pixels of padding to the blurring. Compared to the baseline that is 65.44%. In test 11 the anonymization technique is changed to a Gaussian blur, which shows that the fps achieved is 57.9. That is 48.21% compared to the baseline measurement. 53.6 fps is achieved while also adding 25pixels to the Gaussian blur, this is shown in test 12. This is 44.63% compared to the baseline measurement. Test 13 changes the anonymization technique to masking, which shows that the performance is 82.0 fps. That is equal to 68.28% compared to the baseline. After adding 25pixel padding to the masking technique applied to the detections, the performance recorded is 82.0 fps as shown in test 14. Compared to the baseline that is 68.28%.

The same 14 tests are also conducted on the other hardware platforms. Figure 3 shows the results for the Raspberry Pi 3 platform.

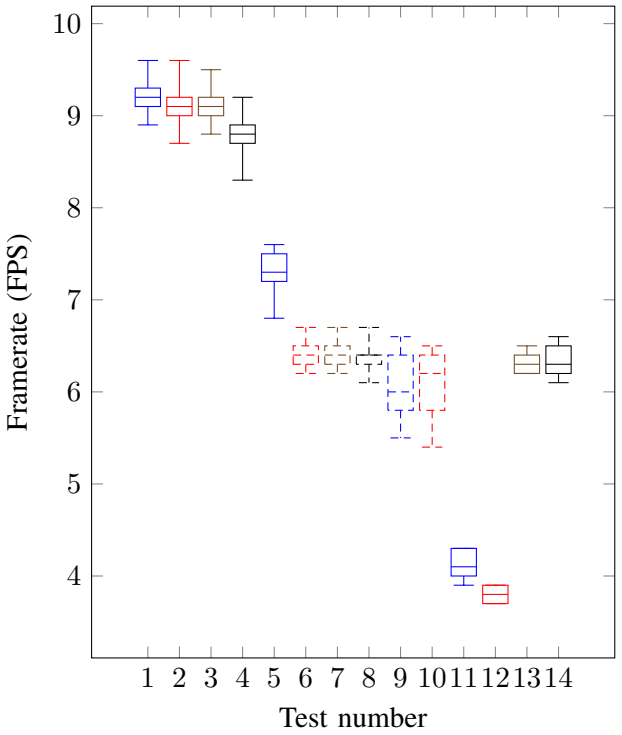

Figure 3: *Hardware setup: Raspberry Pi 3*

The Intel Atom Server results are shown in Figure 4.

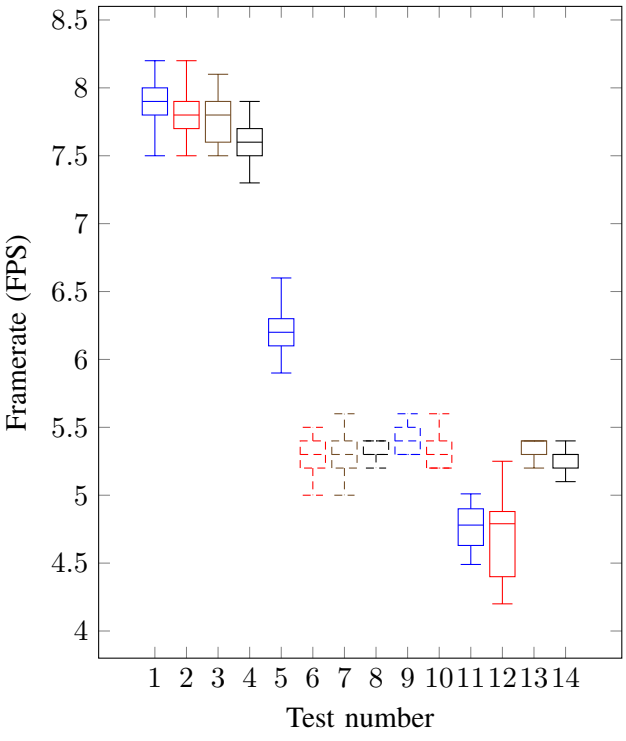

Figure 4: *Hardware setup: Intel Atom Server*

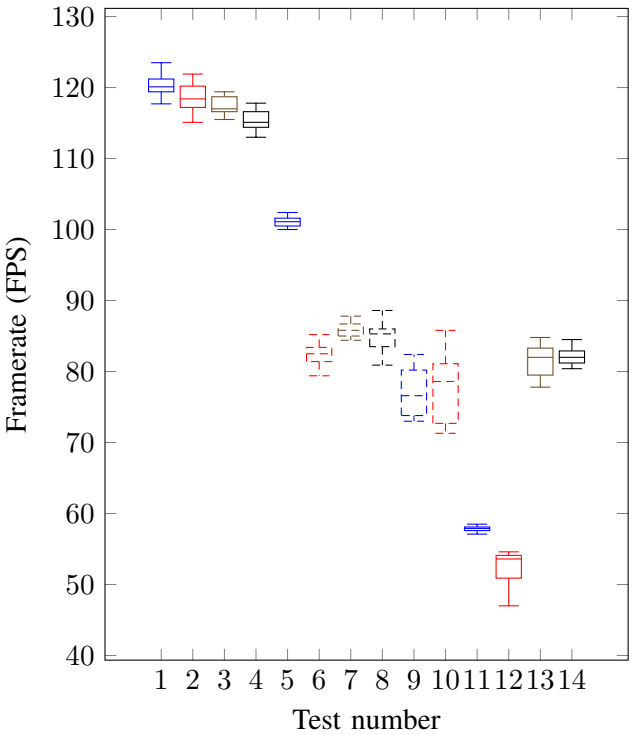

Figure 2: *Hardware setup: Dell desktop*

Similar the results for the Intel Atom Laptop are shown in Figure 5.

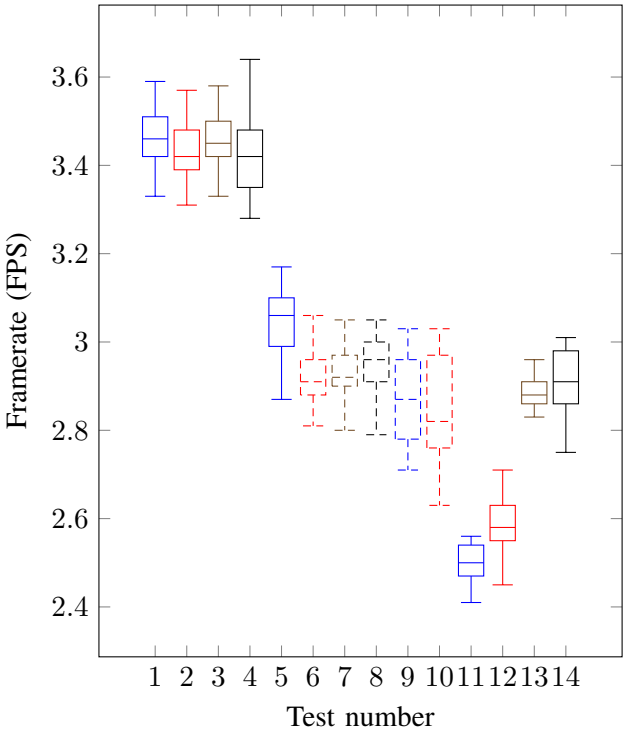

Figure 5: *Hardware setup: Atom laptop*

Table II reflects the performance of each test relative to their respective baseline test (test 1). The results are shown in Table III.

	Dell Desktop (fps)	*Raspberry Pi (fps)*	*Intel Atom Server (fps)*	*Intel Atom Laptop (fps)*
Test 1	100%	100%	100%	100%
Test 2	98.58%	98.91%	98.73%	98.84%
Test 3	97.42%	98.91%	98.73%	99.71%
Test 4	95.84%	95.65%	96.20%	98.84%
Test 5	84.18%	79.35%	78.48%	88.44%
Test 6	68.70%	69.56%	67.09%	84.10%
Test 7	71.44%	69.56%	67.09%	84.39%
Test 8	71.02%	69.56%	67.09%	85.55%
Test 9	63.78%	65.22%	68.35%	82.95%
Test 10	65.44%	67.39%	67.09%	81.50%
Test 11	48.21%	44.56%	60.50%	72.25%
Test 12	44.63%	41.30%	60.63%	74.57%
Test 13	68.28%	68.48%	68.35%	83.24%
Test 14	68.28%	68.48%	67.09%	84.10%

Table III: *Shows the relation between the baseline measurement and the performance hit relative to the baseline measurement*

VI. Discussion

The first thing that became obvious was the overall performance of the Raspberry Pi 3, compared to the two Atom platforms. In general, the Raspberry performed as good or even better than both Atom platforms. This is especially interesting, given the Atom server specs (shown in Appendix IX).

The results clearly show that the Raspberry Pi is not capable of processing the Full-HD video stream at the same rate the IP-camera is putting out the video. Even during the baseline test, the camera is performing around 9.2 fps. To the human eye, this is conceived as stuttering. In test 14, the Raspberry Pi 3 is performing at 6.3 frames per second. Although this performance is far from fluent, the frame rate can still be acceptable, depending on the application. Traditional Video Surveillance systems tend to follow the same behaviour, instead of saving the whole video stream at full frame rate, these systems only save a few frames per second. This way, the storage requirements are far less than in cases where the whole feed is stored.

While running the experiments on the Raspberry Pi 3 and the Intel Atom laptop the temperature of the devices was abnormally high. Initial tests showed that the temperature would reach 80+ degree Celsius. Therefore, we had to take extra cooling measures to prevent crashes or damage to the devices during the experiments. The setup is shown in Appendix XII. Due to the temperature issues, more permanent cooling solutions are needed if the Raspberry Pi 3 or the Intel Atom Laptop are to be used in a production environment.

A key factor influencing the results is the hardware offloading support for encryption and video processing. The Dell desktop supports both AES-128-CBC and H264-codec hardware offloading, while the Raspberry Pi 3 supports only H264 offloading. Both Atom platforms lacked hardware support for cryptographic operations, and therefore had to do all operations in software.

We could not reliably determine if the Intel Atom server used hardware offloading for the video codecs.

Lacking hardware offloading support means that the device has to do these operations in software. Generally, this translates to using more CPU resources, compared to hardware offloading.

Based on the information in the literature, we chose to use three specific anonymization techniques during our experiments; pixelation, Gaussian blurring

and masking. As mentioned before, according to the work of Ruchaud and Dugelay, Gaussian blurring and (normalized box) pixelation are not considered to be secure.e.

However, it is worth noting that they tried to re-identify faces that were anonymized using various levels of Gaussian blur, with a standard deviation ranging from 2 to 8. They stated that even with a standard deviation of 8, they were indeed able to distinctively identify faces. In our tests we used a standard deviation of 25, being the highest setting OpenCV would allow us to use.

The same holds true for the pixelation technique we used. Ruchaud and Dugelay used a maximum box size of 10x10 pixels. We used a grid of 50x50 pixels, a multiple of what they used.

It has yet to be proven that both their statements will still hold for the parameters we used.

Based on our results, the masking filter proved in all cases to be the most resource friendly. All filters cover the same amount of frame real-estate, potentially covering more than necessary. Due to the limitations of the DNN, we were not able to define the ROIs more precisely. Therefore, the ROIs had to be defined as rectangles. Anonymizing the ROIs more precisely would be more intelligent.

VII. Conclusion

Based on the work others did, we concluded that PII extends further than only faces or personal characteristics. PII also includes any information that can directly or indirectly identify a person. Information about what a person is doing can identify the person itself. Therefore, besides filtering persons, we set out to filter computer screens too.

After identifying ROIs, we studied various filtration techniques. Based on related work, we chose three *one-way* filtration techniques; pixelation, Gaussian blurring and masking. Related work showed that *two-way* filtration techniques were either complex, lacked a real implementation or were not reviewed by the cryptographic community.

Our results show that commodity hardware can be used to filter Personally Identifiable Information from real-time video streams. While the results show that the Dell desktop platform is the only hardware platform that is capable of producing a fully anonymized video stream at the same framerate as the IP-camera, other platforms can still be used. The video that is produced by the other platforms is perceived as stuttering. Depending of the use case, this does not have to cause any issues, since typical Video Surveillance Systems often only store a few fps.

The performance hits of each of the tests described in Table III, show that all hardware platforms follow a similar pattern. This indicates that the operations have a simmilar impact on the overall performance. Though a clear relation can be seen between the Dell Desktop and the Raspberry Pi performance hits. The baseline measurements differ as the Dell Desktop can handle 120.1 fps compared to the Raspberry Pi's 9.2 fps. This is only 7.66% of the Dell Desktop baseline.

Due to this relation a prediction model could be devised to predict how much fps a given hardware platform has under a specific test or load.

VIII. Future Work

In this section possible contributions are described that could further increase the efficiency and performance of the setup depicted in this paper.

The effectiveness of the setup is largely defined by the detection model that is being used. Further extending the types of Personally Identifiable Information that could be detected would greatly benefit the real-life implementation and make the setup more versatile.

The impact of using different encryption algorithms with various strengths will further increase the granularity of the results presented in this paper. The same can be said about using different codecs. In our research, we only took the H264 codec into consideration. Therefore, it could be interesting to testing different codecs. This will further increase the granularity of the results depicted in this paper.

The anonymization techniques included in this paper can be further expanded to incorporate other anonymization techniques, as well as testing anonymization techniques on different strength levels.

A possible way to increase the performance of the interception devices is to split the tasks over multiple devices. Distribution or clustering techniques could potentially increase the performance of the setup depicted in this paper. The overclocking capabilities of the Raspberry Pi were not taken into account. Overclocking the device could potentially leverage more performance. But then again, the cooling issues also have to be taken into account.

REFERENCES

[1] T. Piatrik, V. Fernandez, and E. Izquierdo. "The privacy challenges of in-depth video analytics". In: *2012 IEEE 14th International Workshop on Multimedia Signal Processing (MMSP)*. Sept. 2012, pp. 383–386. DOI: 10.1109/MMSP.2012.6343473.

[2] H. Liu, S. Chen, and N. Kubota. "Intelligent Video Systems and Analytics: A Survey". In: *IEEE Transactions on Industrial Informatics* 9.3 (Aug. 2013), pp. 1222–1233. ISSN: 1551-3203. DOI: 10.1109/TII.2013.2255616.

[3] P. Korshunov and T. Ebrahimi. "Using warping for privacy protection in video surveillance". In: *2013 18th International Conference on Digital Signal Processing (DSP)*. July 2013, pp. 1–6. DOI: 10.1109/ICDSP.2013.6622791.

[4] Matthias Huber et al. "A Provably Privacy Preserving Video Surveillance Architecture for an Assisted Living Community." In: *GI-Jahrestagung*. 2014, pp. 563–574.

[5] Nederlandse Spoorwegen. *Meten is weten: realtime reizigers tellen op zes drukke stations (Article in Dutch)*. 2017 (accessed February 1, 2018). URL: http://nieuws.ns.nl/meten-is-weten-realtime-reizigers-tellen-op-zes-drukke-stations/.

[6] Nederlandse Omroep Stichting (NOS). *Reclameborden op A'dam CS weten wanneer en hoelang jij kijkt (Article in Dutch)*. 2017 (accessed February 6, 2018). URL: https://nos.nl/artikel/2191341-reclameborden-op-a-dam-cs-weten-wanneer-en-hoelang-jij-kijkt.html.

[7] "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)". In: *Official Journal of the European Union* L119 (May 2016), pp. 1–88. URL: http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:L:2016:119:TOC.

[8] University of Illinois at Chicago. *Data Classifications - Information Security and Privacy Office*. 2017 (accessed Januari 16, 2018). URL: https://security.uic.edu/data-classifications/.

[9] Jehan Wickramasuriya et al. "Privacy protecting data collection in media spaces". In: *Proceedings of the 12th annual ACM international conference on Multimedia*. ACM. 2004, pp. 48–55.

[10] Cesar Pantoja, Virginia Fernandez Arguedas, and Ebroul Izquierdo. "Anonymization and De-identification of Personal Surveillance Visual Information: A Review". In: ().

[11] Maneesh Upmanyu et al. "Efficient privacy preserving video surveillance". In: *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE. 2009, pp. 1639–1646.

[12] Natacha Ruchaud and Jean-Luc Dugelay. "Automatic Face Anonymization in Visual Data: Are we really well protected?" In: *Electronic Imaging* 2016.15 (2016), pp. 1–7. ISSN: 2470-1173. DOI: doi:10.2352/ISSN.2470-1173.2016.15.IPAS-181. URL: http://www.ingentaconnect.com/content/ist/ei/2016/00002016/00000015/art00014.

[13] Sk Md Mizanur Rahman et al. "A real-time privacy-sensitive data hiding approach based on chaos cryptography". In: *Multimedia and Expo (ICME), 2010 IEEE International Conference on*. IEEE. 2010, pp. 72–77.

[14] J. Cichowski and A. Czyzewski. "Reversible video stream anonymization for video surveillance systems based on pixels relocation and watermarking". In: *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. Nov. 2011, pp. 1971–1977. DOI: 10.1109/ICCVW.2011.6130490.

[15] Swann Scholtes and Chris Kuipers. *RP1 - VideoStreamFiltration - Proof of Concept code*. Jan. 2018. URL: https://gitlab.os3.nl/ckuipers/RP1-VideoStreamFiltration.

[16] Jeff Donahue. *BAIR/BVLC CaffeNet Model*. 2017 2017 (accessed January 12, 2018). URL: https://github.com/BVLC/caffe/tree/master/models/bvlc_reference_caffenet.

Appendices

IX. Hardware specifications

Description	Model	Dell Optiplex 7010
	Type	Desktop
CPU	CPU	Intel i5-3570S
	Architecture	x86_64
	Clockspeed	3.1GHz
	Cores/Threads	4c/8t
Memory	Type	DDR3
	Size	12GB
	Speed	1600 MHz
Chipset	Graphics card	Intel HD Graphics
	Chipset	
	Speed	
Storage	Type	Solid State Disk
	Model	Samsung SM84
	Size	128GB
Networking	NICs	1x
	Speed	1 Gbps
Software	OS	Ubuntu Desktop 16.04.3 LTS
	Libraries	Python 3.5.1
		OpenCV 3.3.0

Table IV: Dell Desktop hardware specifications

Description	Model	Raspberry Pi3 model B
	Type	System on a Chip (SoC)
CPU	CPU	Broadcom BCM2837
	Architecture	ARMv8
	Clockspeed	1,2 GHz
	Cores/Threads	4c
Memory	Type	DDR2
	Size	1GB
	Speed	400MHz
Chipset	Graphics card	VideoCore IV
	Chipset	Broadcom BCM2837
	Speed	400MHz
Storage	Type	SD Card
	Model	SanDisk
	Size	16GB
Networking	NICs	1x
	Speed	100 Mbps
Software	OS	Raspbian 9.3
	Libraries	Python 3.5.1
		OpenCV 3.3.0

Table V: Raspberry Pi 3 hardware specifications

Description	Model	Kimsufi KS-1
	Type	Server
CPU	CPU	Intel Atom N2800
	Architecture	x86_64
	Clockspeed	1.86GHz
	Cores/Threads	2c/4t
Memory	Type	DDR3
	Size	2GB
	Speed	1066 MHz
Chipset	Graphics card	
	Chipset	
	Speed	
Storage	Type	Magnetic Hard Disk
	Model	Western Digital
	Size	500GB
Networking	NICs	1x
	Speed	100 Mbps
Software	OS	Ubuntu Server 16.04.3 LTS
	Libraries	Python 3.5.1
		OpenCV 3.3.0

Table VI: Intel Atom Server hardware specifications

Description	Model	Samsung NC10
	Resolution	Netbook
CPU	CPU	Intel Atom N270
	Architecture	x86
	Clockspeed	1.6GHz
	Cores/Threads	2c
Memory	Type	DDR2
	Size	1GB
	Speed	1066 MHz
Chipset	Graphics card	Intel GMA 950
	Chipset	Intel 945GSE
	Speed	
Storage	Type	SSD
	Model	MX300
	Size	275GB
Networking	NICs	1x
	Speed	100 Mbps
Software	OS	Ubuntu Server 16.04.3 LTS
	Libraries	Python 3.5.1
		OpenCV 3.3.0

Table VII: Intel Atom Laptop hardware specifications

Description	Model	Ubiquiti UVC-G3-DOME camera
	Resolution	1920x1080 (Full HD)
	FPS	30fps
Networking	NICs	1x
	Speed	1Gbps
	Video protocol	RTSP

Table VIII: Ubiquiti IP-camera specifications

X. Deep Neural Network model accuracy

The model is a snapshot of iteration 310.000. The statistics that are available are about iteration 313.000.

Description	Validation accuracy	51,412%
	Loss	1,82328
	Top-1 accuracy	57,4%
	Top-5 accuracy	80,4%

Table IX: DNN model accuracy at iteration 313.000

XI. Test setup

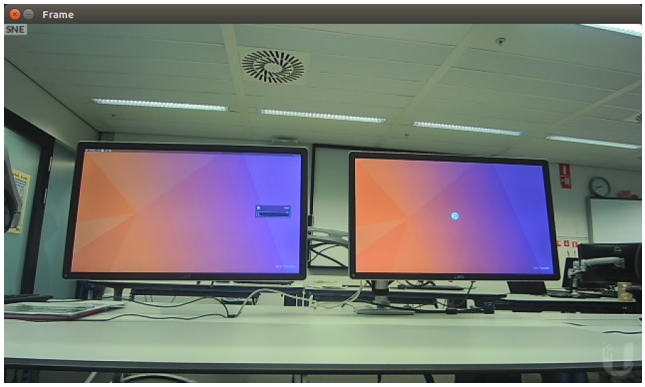

Figure 6: Baseline

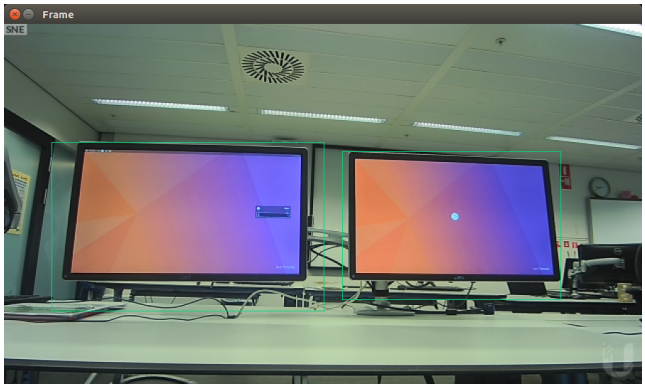

Figure 7: Boxes

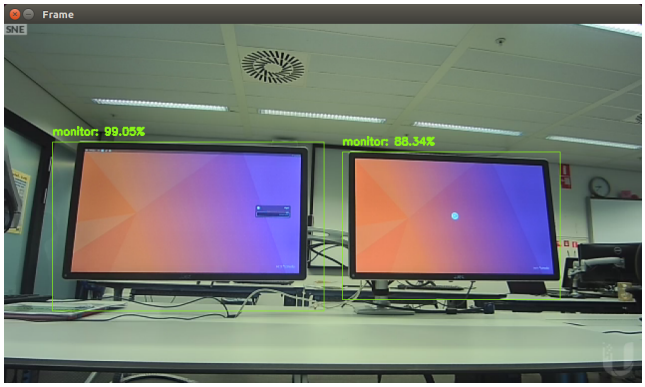

Figure 8: Labels

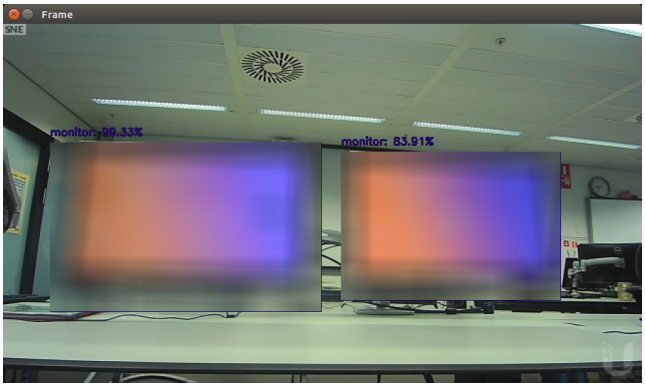

Figure 9: Blur

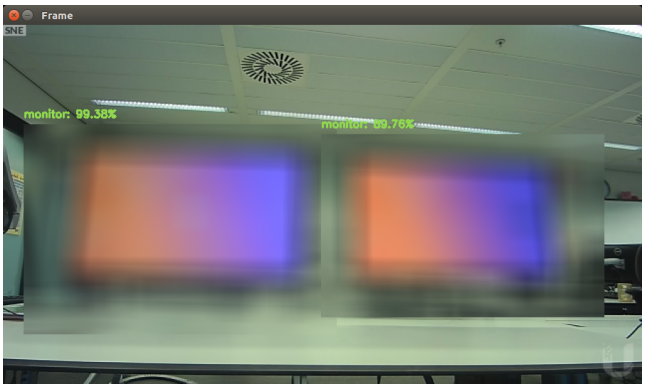

Figure 10: Blur padding

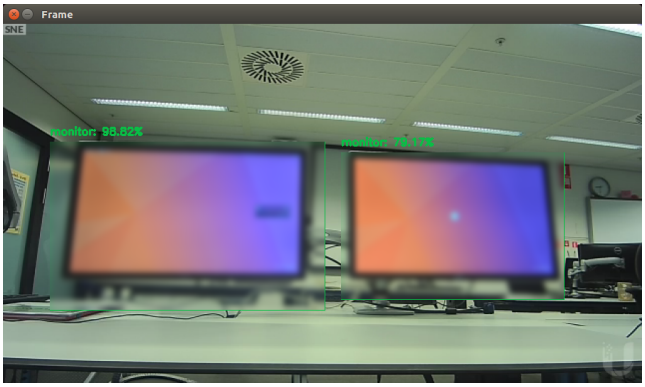

Figure 11: Guassian blur

Figure 12: Gaussian padding

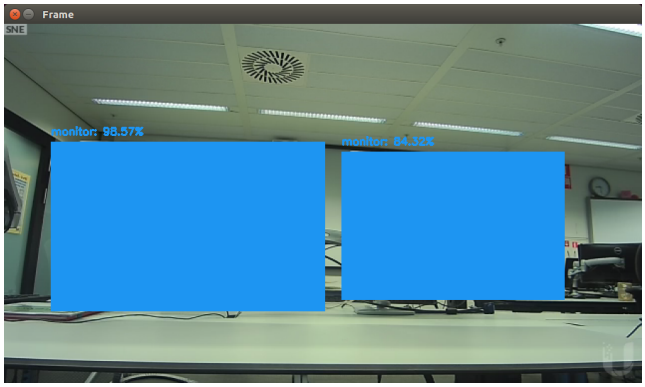

Figure 13: Masking

XII. Practicalities

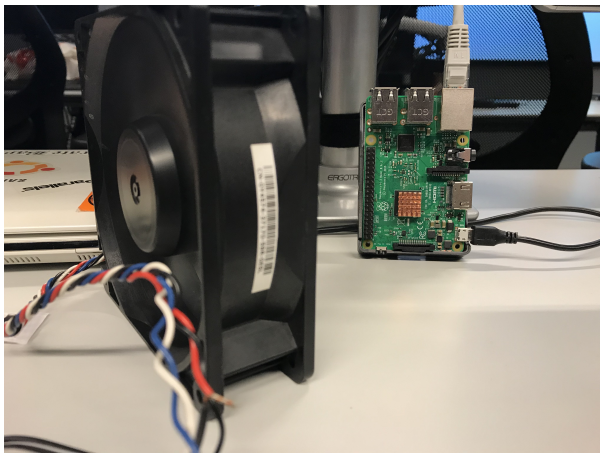

Figure 14: Raspberry Pi 3 extra cooling