Boosting the Performance of the Neural Network Using Symmetry Properties for the Prediction of the Shower Maximum Using the water-Cherenkov Detectors of the Pierre Auger Observatory as an Example

Steffen Hahn,[*] **Markus Roth, David Schmidt, and Darko Veberič**

Institute for Astroparticle Physics, Karlsruhe Institute of Technology KIT, Germany
E-mail: steffen.hahn@kit.edu

To probe physics beyond the scales of human-made accelerators with cosmic rays demands an accurate knowledge of their primary mass composition. Using fluorescence detectors, one is able to estimate the mass by measuring the depth of the shower maximum $X_{\max}$. These, however, exhibit a very low duty cycle of typically below 15%.
Inferring $X_{\max}$ from a surface detector array (SD) such as the water-Cherenkov array of the Pierre Auger Observatory is highly non-trivial due to the inherent complexity and fluctuations of the shower footprint. Moreover, the sheer amount of data makes it non-trivial to find hidden patterns in the spatial and temporal distributions of detector signals. Neural networks provide a straightforward way of tackling such a problem doing a data-driven analysis.
Relying solely on geometrical quantities, timing, and the signal-time information of the SD stations, we show that by exploiting the symmetries due to their triangular arrangement, we are able to boost a standard analysis network significantly without modifying its architecture or training process. Furthermore, these considerations yield a standardization procedure which also enables us to encode the footprint information in a memory-efficient way. The presented procedure can also be generalized and extended to systems whose setup has an underlying hexagonal geometry.

37th International Cosmic Ray Conference (ICRC 2021)
July 12th – 23rd, 2021
Online – Berlin, Germany

[*]Presenter

1. Introduction

Studying Ultra-High Energy Cosmic Rays (UHECRs) is – as today – the only way to probe particle physics beyond the limitations of human-built accelerators [1] and to explore the most extreme processes in the Universe. Due to their highly suppressed energy spectrum [2], direct detection methods are unfeasible. Therefore, today's state-of-the-art approaches focus on the analysis of the cascade of secondary particles (air shower) emerging from the interactions of an UHECR with our atmosphere. One way of indirect analysis is by measuring a slice of the shower development plane at ground level. This shower footprint contains complex non-trivial information that makes it possible to classify showers without the direct need of longitudinal information of the shower. The World's largest observatory following this indirect approach is the Pierre Auger Observatory [3, 4]. It measures time signals of the shower footprint by using a grid of isometric triangular arranged water-Cherenkov detectors: the Surface Detector (SD). In addition, it uses Fluorescence Detectors (FD) to probe the longitudinal shower development which allows a cross-calibration with the SD.

To gain an understanding of the involved high-energy physics, knowledge about the composition of the UHECRs is required [5]. One way to obtain this is by estimating the depth of the shower maximum X_{max}. Being the point of maximum deposit of calorimetric energy it is directly measurable by the FDs. The FDs, however, have only a limited uptime of below 15% [4]. Hence, the natural question arises if a SD can be used for a X_{max} prediction. Using deep neural networks this seems possible [6]; even for real data [7].

The isometric triangular structure of the surface detector of the Pierre Auger Observatory possesses intrinsic symmetries. To first-order[1] we are able transform the shower footprint (measured by SD) in the surface detector plane into a particularly chosen triangle without changing the topology of the corresponding shower-footprint. Models that use the shower-footprint have either to "learn" this during the fitting/training process or encode it in some way. In this work, we show that by using a shower footprint standardization technique and taking advantage of the underlying symmetries of our initial data, the predictive power of models can be improved. Additionally, we show that this method enables us to work memory efficient which is advantageous for models with large amounts of input data, such as models based on neural networks. This method is universal. That is, it may be applied to any detector geometry having similar structures.

In Section 2, we derive the procedure used to standardize our shower footprints and show the additional standardization used for our input data. Afterwards, in Section 3, we describe the used neural network and the training process that yielded the results we present in this work. In Section 4, we discuss the effect of the procedure and further benefits. Finally, we conclude the work and give a further outlook in Section 5.

2. Standardization procedure

The complete standardization of our data follows two steps. First, we perform a "geometrical" standardization. That is, we transform the shower footprints in such a way that all of there corresponding (to-the-ground-projected) shower axes point in an azimuth interval of 30°. Afterwards,

[1]By ignoring Earth's magnetic field.

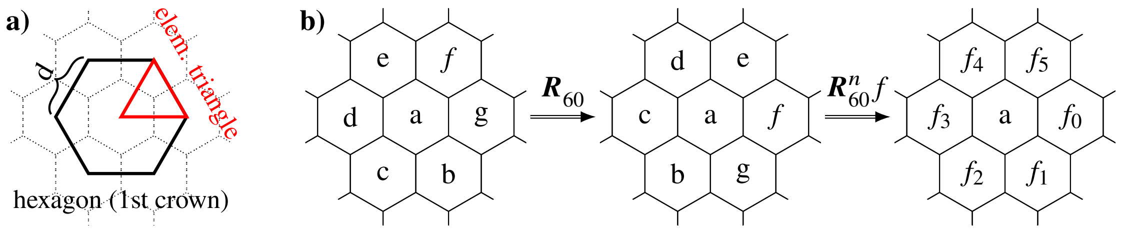

Figure 1: Illustration of first crown (**a**) and of the rotations defined in Eq. (1) (**b**). The center point of the rotation is at the grid point which lies in the cell "a" and typically includes the shower core. Applying the rotation matrix $\boldsymbol{R}_{60}$ on the left grid rotates all stations clockwise around "a". Following only the cell f, we are able to show the effect of all other rotation matrices (see Eq. (1)). In the right grid, the index of f corresponds to the used matrix n.

we standardize the data itself to allow for a smoother training process of our neural network (see Section 3.1).

2.1 Shower footprint standardization

The SD of the Pierre Auger Observatory is an almost ideal isometric triangular grid. Hence, we may describe the positions of its stations by $\boldsymbol{d}_i = x_i\boldsymbol{u} + y_i\boldsymbol{v} \equiv [x_i(1,0)^\intercal + y_i(-1/2, \sqrt{3}/2)^\intercal]d$, where $\boldsymbol{u}$ and $\boldsymbol{v}$ are a particular choice of base vectors for the triangular lattice. The length d is the distance between lattice points. We choose $d = 1500\,\mathrm{m}$ which corresponds to the approximate spacing of the regular Auger SD. The integer coordinates x_i and y_i uniquely identify each lattice point and, therefore, each WCD station of the array.

For our standardization procedure, we want to find all distance-conserving transformations that preserve an infinite lattice around a fixed lattice point. This fixed lattice point will correspond to the station closest to the intersection point of the shower axis and the detector plane. We define that a station is in crown m if we only would need m jumps between neighboring stations to reach this fixed, central station (see Fig. 1).

Intrinsically, a triangular grid is invariant under rotations of multiples of 60°. Using our choice of integer coordinates, we express the 60° rotation as

$$\begin{pmatrix} x_i' \\ y_i' \end{pmatrix} = \boldsymbol{R}_{60} \begin{pmatrix} x_i \\ y_i \end{pmatrix} \equiv \begin{pmatrix} 0 & 1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix}. \tag{1}$$

The matrix $\boldsymbol{R}_{60}$ rotates all of our grid points 60° clock-wise (see Fig. 1). Therefore, the n-th "grid-conserving" rotation matrix is $\boldsymbol{R}_{60n\,(\mathrm{mod}\,360)} = (\boldsymbol{R}_{60})^n$, whereby $\boldsymbol{R}_0 = \mathbf{1}$. To gain a better understanding of these matrices, we depict their transformations in Fig. 1.

In addition to the rotational symmetries, the reflections corresponding to the reflection symmetries of a hexagon also conserve the grid. Fortunately, we require only one of them since all other reflections can be constructed from the rotations and one reflection. Therefore, from a general point of view, the choice of this axis is arbitrary. In this work, we choose the straight line defined by $\boldsymbol{u} + \boldsymbol{v}$ (see Eq. (1)) because it provides us with the simple reflection matrix for our choice of unit

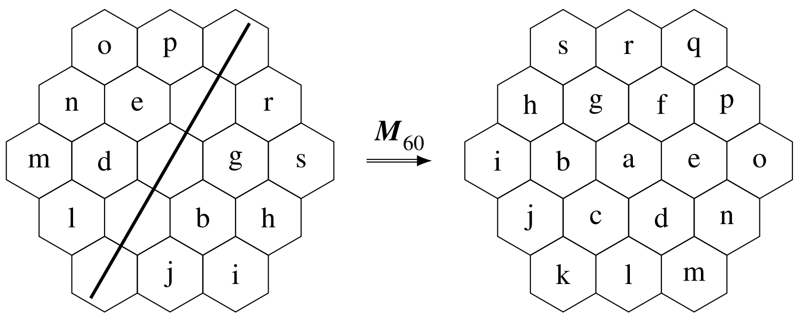

Figure 2: Illustration of reflection defined in Eq. (2) on the first and second crown. The reflection axis which is connected to $\boldsymbol{M}_{60}$ is defined by $\boldsymbol{u} + \boldsymbol{v}$.

vectors,

$$\begin{pmatrix} x_i' \\ y_i' \end{pmatrix} = \boldsymbol{M}_{60} \begin{pmatrix} x_i \\ y_i \end{pmatrix} \equiv \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix}. \tag{2}$$

We illustrate the effect of $\boldsymbol{M}_{60}$ in Fig. 2.

Since reflections and rotations are the only isometries in the two dimensional plane, we only have to show that we are able to construct all other reflections with the matrices of Eq. (1) and Eq. (2). Then, our choice of transformations covers all possible symmetries in a triangular grid.

We denote the six distinct reflection operations as $\boldsymbol{M}_0, \boldsymbol{M}_{30}, \ldots \boldsymbol{M}_{150}$ where the subscript corresponds to the angle of the reflection axis with respect to the (horizontal) x-axis. Since both $\boldsymbol{M}_0$, $\boldsymbol{M}_{60}$, $\boldsymbol{M}_{120}$ and $\boldsymbol{M}_{30}$, $\boldsymbol{M}_{90}$, $\boldsymbol{M}_{150}$ lie 60° apart we are able to construct them via the rotation matrices. As a consequence, we only have to connect those both groups of matrices. Fortunately,[2] this can be done by the relation $\boldsymbol{M}_0 = \boldsymbol{R}_{60}\boldsymbol{M}_{30}$. Since we have twelve unique transformations, we are able to rotate and reflect any shower footprint into a pre-defined azimuth range of 30° by applying the transformation matrices on the integer coordinates. Consequentially, this reduces the phase space by a factor of twelve.

2.2 Representation of shower footprint in memory

Since the triangular structure demands that each grid point has at least six neighbors it is non-trivial to choose an efficient representation of the shower footprint in computer memory. We have opted for an one-to-one integer coordinate mapping choosing the central station as the origin (see Fig. 3). This kind of representation exhibits stations belonging to non-closed crowns in the north-west and south-east region. Depending on the choice of total memory-grid size these "border regions" of the shower footprint consist of $m^2 + m$ elements where m is the maximum crown still contained in the memory.

In the case of raw footprints the showers are distributed uniformly in azimuth. Therefore, we can only increase information in memory by adjusting the grid size accordingly. However, the gain of information is decreasing with increasing memory-grid size. Since shower footprints of inclined showers are elongated along the projected shower axis, we are able to use the standardization procedure in Section 2.1 to circumvent this by maximizing the information content in the border

[2] $\boldsymbol{M}_{30} = \boldsymbol{R}_{-30}\boldsymbol{M}_0\boldsymbol{R}_{30} \Rightarrow \boldsymbol{R}_{60}\boldsymbol{M}_{30} = \boldsymbol{R}_{30}(\boldsymbol{M}_0\boldsymbol{R}_{30}\boldsymbol{M}_0)\boldsymbol{M}_0 = \boldsymbol{R}_{30}\boldsymbol{R}_{-30}\boldsymbol{M}_0 = \boldsymbol{M}_0$.

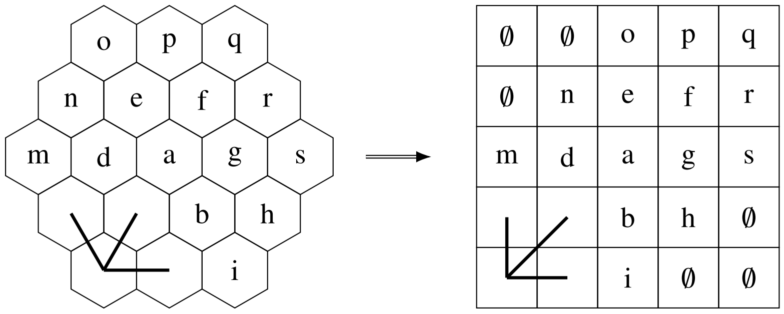

Figure 3: Memory layout (right) of a triangular grid up to the second crown (left). The south-east and north-west corners of the memory map contains (in this case) no data (marked by ∅) which is a consequence of the different number of neighbors in the hexagonal and square grid. These would be part of the third and fourth crown. In this representation, the number of these "empty data points" for crown m is the difference of hexagons inside one crown compared to $(2m + 1)^2$ which is the size of a square lattice fitting all crowns: Hence, the memory map has $m^2 + m$ elements that correspond to not complete crowns.

regions for a given memory-grid size. To do this we have to align the transformed shower axes with the axis in memory that goes trough the memory regions of incomplete crowns (represented by the ∅ symbol in Fig. 3)).

As a consequence, to show that the shower footprint standardization introduced in Section 2.1 boosts the predictive power of footprint based methods itself, we have to ensure that the boost itself is not due to an increase of signal density. Therefore, for this analysis we choose to set all parts of the footprint that do not belong to complete crowns in the memory to zero (cf. Fig. 3) nullifying this additional advantage.

2.3 Input data standardization

In addition to the geometrical standardization of our shower footprint, we also perform various standardizations on our chosen input data. We use the average time signals $S(t)$ of the water-Cherenkov tanks together with timing information and trigger masks as inputs [6]. All of this information is embedded into multi-dimensional arrays which encode the shower footprint as mentioned in Section 2.2. We trim the time signals $S(t)$ to an (arbitrary) number of 120 time bins starting from the bin at which the corresponding detector has triggered. Afterwards the new traces are normalized by $\hat{S}(t) = \lg\left(1 + S(t)/\mathrm{VEM}\right)/\lg\left(1 + 100\right)$. During the standardization of the time signal we loose the relative time between single stations, we use this timing information relative to the central station as another input. We standardize this by dividing by the standard deviation of the relative times of the whole dataset. The trigger mask is just a boolean array which is `True` if a position corresponds to a triggered station.

Furthermore, we standardize our output X_max using $\hat{X}_\mathrm{max} = (X_\mathrm{max} - \langle X_\mathrm{max}\rangle)/\sigma_{X_\mathrm{max}}$, where $\langle\cdot\rangle$ and σ are the mean and standard deviation of the X_max of our training data set, respectively. We use the Gaisser-Hillas X_max obtained by a fit to the longitudinal profile of energy deposit in our simulation (see Section 4). Since our raw dataset contains some extremely high-valued outliers we perform the quality cut $X_\mathrm{max} < 1200\,\mathrm{g/cm^2}$. Including them in the training process is causing convergence problems.

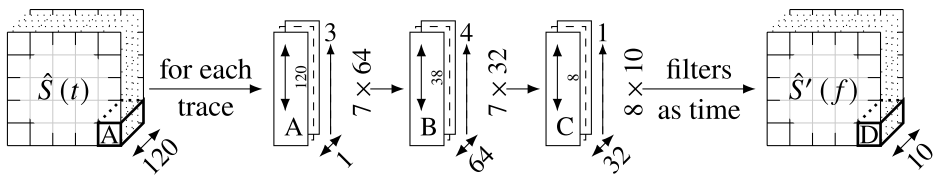

Figure 4: Architecture of trace analyzer. The network uses the same weights and layers for each trace. In total there are three convolution layers which gradually reduce the number of time bins by using valid convolutions. The $n \times m$ to the right of each layer represents the convolution window size n and number of used filters m. The number above the vertical arrow indicates the used stride. We use the filters – in the end – as new "time" bins.

We chose to restrict the shower footprint to the fifth crown. That is that our input data for each event consists of one $11 \times 11 \times 120$ tensor and an $11 \times 11 \times 2$ tensor.

3. Neural Network architecture and training

For all of our predictions, we rely on one feed-forward network architecture. We use "vanilla" Tensorflow [8] for the implementation. Therefore, the training procedure is based on a standard back-propagation. The updates of the layer weights are based on loss-dependent (micro-batch) updates.

3.1 General design

The network architecture is taken from [6]. It consists of (roughly) three different logical units: a time signal analyzer (i), a spatial analyzer (ii), and a final prediction unit (iii). We sketched (i) and (ii) in Fig. 4 and Fig. 5, respectively.

The first part (i) is a "parallel" neural network that extracts the 10 feature bins from our 120 time bins. Therefore, we call these time signal features. The network is "parallel" in the sense that it treats every time signal exactly the same. All of these see the same sub-network which is composed out of a mix of different convolutional layers (see Fig. 4). Each of these layers uses a ReLu function which is not represented in the graphic. Afterwards, the trace features are concatenated to our other inputs (see Section 2.3). These are timing information given by the detector stations and the activity map. Hence, at this point we have a $11 \times 11 \times 12$ array which is used as an input of the second part (ii) of the network. The spatial analyzer is a multi-layered convolutional network that connects the spatial coordinates via 3×3 filters (see Fig. 5). To improve the training performance this sub-network uses skipping connection. For each element the original input is concatenated together with the output of a two-dimensional convolutional layer that has the same number of filters as the (time-)length of the input. Hence, after each of those the length doubles and subsequently a higher number of filters has to be used. As last layer we use a convolutional layer with 192 filters. In the final step (iii), we have chosen the simplest design choice. We flatten the output of (ii) and using it as an input for a dense layer with one unit which is the predictor for $X_{\max}$. In addition, during training we use a dropout layer setting 30% of (iii)'s inputs randomly to zero to prevent over-fitting.

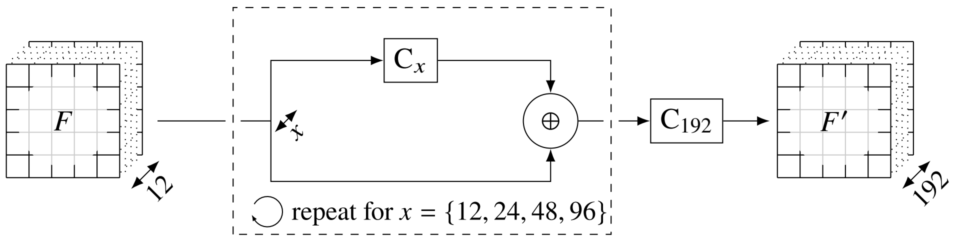

Figure 5: Architecture of spatial analyzer. Again, the networks uses a multi-layer, convolution architecture. In contrast to Fig. 4, however, the filters have a spatial extend. Hence, they connect the different space bins in our memory map (see Fig. 3). No strides are used and for each filter the padding is set to same to prevent the memory from shrinking. The network uses a form of skipping connections. That is in each sub-layer the original input is concatenated to the output of a convolutional layer that uses the same number of filters as the time bins doubling them for each subsequent layer. This process is repeated until the output size is 192. After another convolutional layer we end up with a $11 \times 11 \times 192$ output tensor.

3.2 Training procedure

We split our data set (120,000 events) into two parts: One for the training (80%) and one for the evaluation of its performance (20%). To control the training process and prevent over-fitting we use a small part of the training set as a validation set. After each training epoch, we evaluate the performance of the network with the latter. It is not used for the fitting procedure itself and, moreover, not used in the testing phase. This enables us to check for abnormalities during the training process. We use a L2 loss function and a batch-sizes of 32 elements for the training. Our network stops the training process when the value of the validation loss does not improve for six epochs or suddenly diverges. We use the optimizer Adam [9] with an initial learning rate of 0.0032. The maximum number of epochs is 100. When the learning rate reaches a plateau (for four epochs) the learning rate is reduced by a factor of 0.7.

4. Application on simulation data

To show the boost because of the geometric standardization it is sufficient to use only a subset of simulated showers and keep the comparison as simple as possible. Therefore, we only use protons air showers subject to the hadronic model QGSJet-II.04 [10]. We use a air shower library simulated with Corsika [11] as the basis of our training, validation, and test set. Each shower is used ten times for our detector simulation [12]. To counteract deviations due to different local minima in the solution space we train 40 models and draw from those forty results forty times three randomly. Then from each of the three we take the best performing one.

Doing a training with a standardized data set provides an visible improvement to our test prediction (see Fig. 6) over the entire investigated energy range reducing the std by $5\,\mathrm{g cm^{-2}}$ on average while not significantly changing the bias.

5. Discussion and summary

We conclude that using this procedure enables us to brush aside the intrinsic "degeneracy" due to all underlying symmetries of the triangular grid. It reduces the original phase space approximately

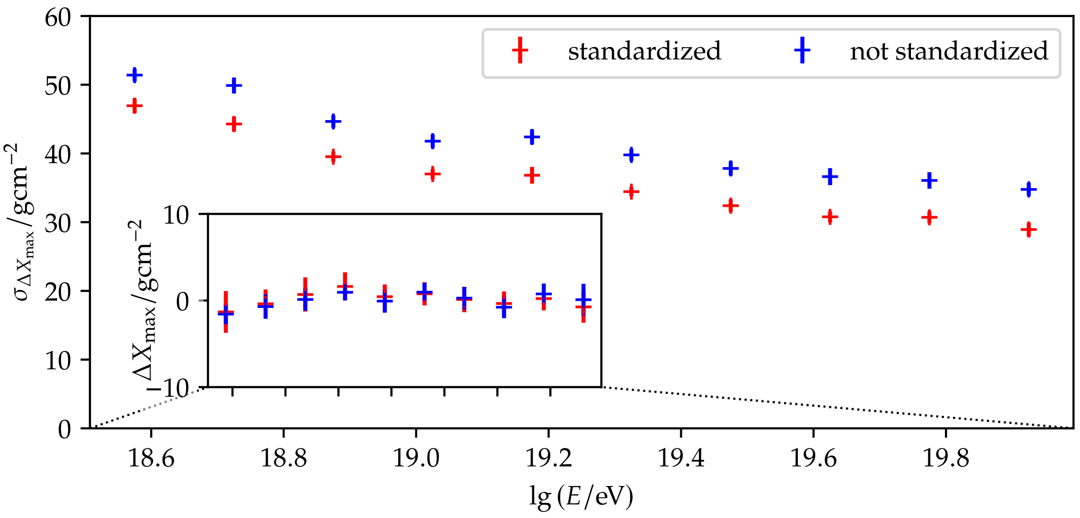

Figure 6: Comparison of standard deviation of $\Delta X_{\max}$ of not standardized (blue) and standardized (red) shower footprint We obtain the errors by a bootstrap procedure which is described in Section 4. We find that for all energy bins the new procedure yields better results. Furthermore, looking at the error bins, the plot indicates that the procedure is also slightly more stable.

to a twelfth of the original while introducing no major disadvantages when the shower axis is well defined. This is advantageous for a analysis based on a neural network if compared to regular shower footprints. Moreover, we demonstrated that by applying the standardization in the right way the data inside of the chosen memory map can be maximised.

The algorithm described (or variations of it) is suitable for a number of applications which have specific symmetries given, e.g., FD, CTA or HESS with their "hexagonal" camera structure (even if some symmetry is broken in these applications by the preferred direction of the air showers).

References

[1] R. Alves Batista, J. Biteau, M. Bustamante, K. Dolag, R. Engel, K. Fang et al., *Front. Astron. Space Sci.* **6** (2019) 23.

[2] The Pierre Auger Collaboration, *Phys. Rev. Lett.* **125** (2020) .

[3] The Pierre Auger Collaboration, *Nucl. Instrum. Meth. A* **798** (2015) 172.

[4] The Pierre Auger Collaboration, *arXiv:1604.03637* (2016) .

[5] K. Greisen, *Annu. Rev. Nucl. Sci.* **10** (1960) 63.

[6] J. Glombitza, N. Eich, M. Erdmann and L. Geiger, *Verh. Dtsch. Phys. Ges.* (2018) .

[7] The Pierre Auger Collaboration, *arXiv:2101.02946* (2021) .

[8] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean et al., *Tensorflow: A system for large-scale machine learning*, in *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pp. 265–283, 2016.

[9] D.P. Kingma and J. Ba, *arXiv:1412.6980* (2014) .

[10] S. Ostapchenko, *Nucl. Phys. B, Proc. Suppl.* **151** (2006) 143.

[11] D. Heck, J. Knapp, J. Capdevielle, G. Schatz, T. Thouw et al., *Report fzka* **6019** (1998) .

[12] S. Argiro, S. Barroso, J. Gonzalez, L. Nellen, T. Paul, T. Porter et al., *Nucl. Instrum. Meth. A* **580** (2007) 1485.